VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

RESOLUCIÓN AUTOMÁTICA DE UNA PRUEBA DE RAVEN MEDIANTE RAZONAMIENTO DIAGRAMÁTICO

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:

EMMANUEL HEGMANN GONZALEZ

DIRECTOR: DR. LUIS ALBERTO PINEDA CORTÉS

MÉXICO, D.F., 2012

Agradecimientos

A los Doctores Luis Pineda, Humberto Carrillo y Fernando Arámbula. A mis sinodales, los Doctores Ana Lilia Laureano, Iván Meza, Carlos Gershenson y Francisco Hernández. A mis profesores. Al personal del posgrado. A mi familia. Gracias por su apoyo, orientación y paciencia.

Así mismo, agradezco a los siguientes programas por su apoyo económico a este proyecto:

- Consejo Nacional de Ciencia y Tecnología, Becas Nacionales. CVU 254155, beca 231967.
- Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica. Proyecto IN-115710, bajo la responsabilidad del Dr. Luis Alberto Pineda Cortés.

Emmanuel Hegmann Gonzalez
Enero de 2012

Resumen

La prueba de matrices progresivas de Raven (RPM, por sus siglas en inglés) es una serie de problemas en los que el participante debe elegir la opción que complete mejor un patrón visual. La Figura 1 muestra un problema similar a los de la prueba.

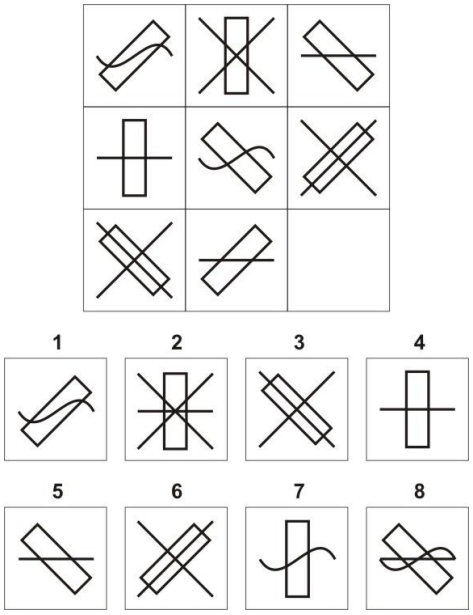


Figura 1. Problema similar a los de la prueba.

La prueba busca medir la habilidad de las personas para descubrir relaciones, y es conocida como una prueba de *cociente intelectual* (IQ). Aunque su validez para "medir la inteligencia" es un tema polémico, en el campo de la inteligencia artificial representa una tarea interesante para poner a prueba modelos computacionales de razonamiento.

En este proyecto se desarrolló un programa de cómputo para resolver la RPM de forma automática, tomando como base una teoría para la síntesis de conceptos geométricos. Partiendo de una representación simbólica de la prueba, el programa ha permitido investigar la aplicabilidad de esta teoría para resolver problemas de razonamiento abstracto. El programa se enfocó en modelar el proceso de inducción de reglas en la prueba estándar y logró resolver 54 de los 60 problemas, superando el puntaje de los trabajos anteriores. Además, el programa fue capaz de resolver problemas más generales que los de la RPM. Este proyecto requirió analizar las características de la RPM, los procesos necesarios para su resolución y los procesos involucrados en la inteligencia.

Palabras clave: prueba de matrices progresivas de Raven, síntesis de conceptos geométricos, esquemas de acción, razonamiento inductivo, analogía visual, simulación cognitiva.

Índice

Índice de figuras

Índice de tablas

1 Introducción

En este proyecto se desarrolló un programa de cómputo para resolver automáticamente una prueba de Raven. Este capítulo presenta los antecedentes del problema.

La Sección 1.1 describe la prueba de Raven y su relevancia en el campo de la psicología. En la Sección 1.2 se exponen seis estudios en los que se han implementado programas para resolver la prueba. La Sección 1.3 presenta una teoría para la síntesis de conceptos geométricos que se tomó como base para desarrollar este proyecto. La Sección 1.4 retoma lo expuesto anteriormente para definir los objetivos de este trabajo. Finalmente, la Sección 1.5 presenta un resumen del capítulo.

El Capítulo 2 presentará el modelo propuesto en este trabajo, así como sus resultados. El Capítulo 3 expondrá las conclusiones que este proyecto permitió obtener, así como varios temas abiertos para trabajos futuros.

1.1 La prueba de Raven

La prueba de matrices progresivas de Raven (o RPM, por sus siglas en inglés) es un conjunto de problemas en los que el participante debe elegir la opción que complete mejor un patrón visual (Raven et al., 2004). Es conocida como una prueba de *cociente intelectual* (IQ) y contiene tres tipos de problemas[1]:

- Problemas en los que el patrón a completar es una sola imagen y se presentan 6 posibles respuestas (Figura 1.1).
- Problemas en los que el patrón es una matriz de 2x2 y se presentan 6 posibles respuestas (Figura 1.2).
- Problemas en los que el patrón es una matriz de 3x3 y se presentan 8 posibles respuestas (Figura 1.3).

Además existen varias versiones de la RPM, cada una con diferentes problemas y niveles de dificultad:

- La versión estándar (o SPM) contiene 60 problemas en 5 series (etiquetadas de la A a la E, y con 12 problemas en cada serie) y está impresa en tinta negra sobre papel blanco.
- La versión a color (CPM) contiene 36 problemas en 3 series (etiquetadas A, Ab y B, con 12 problemas en cada serie) y está diseñada para niños, personas mayores o personas con discapacidades mentales.
- La versión avanzada (APM) contiene 48 problemas en dos series (etiquetadas I y II, con 12 y 36 problemas en cada serie), está impresa en tinta negra sobre papel blanco, y está diseñada para diferenciar mejor a las personas que obtienen un alto puntaje en la prueba estándar.

[1] Por la confidencialidad de las pruebas, en este documento no se muestran problemas originales, sino adaptaciones de los mismos.

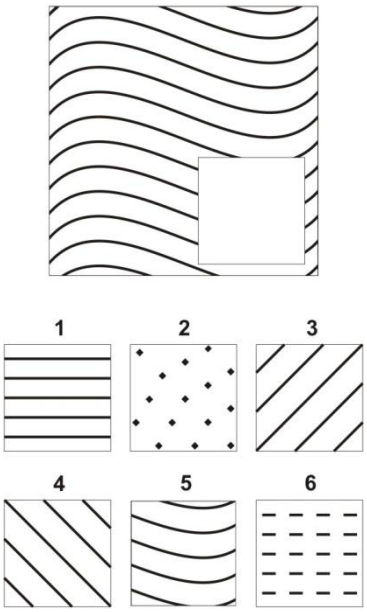


Figura 1.1. Problema con una sola imagen. La respuesta correcta es 5.

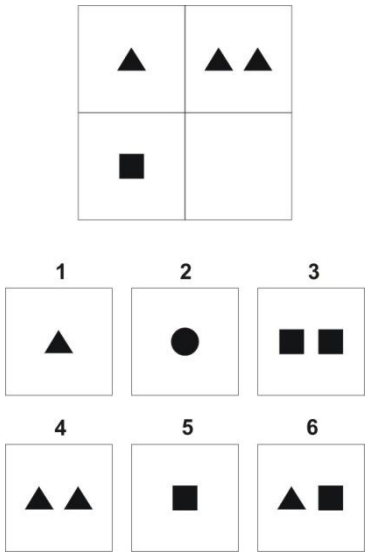


Figura 1.2. Problema con matriz de 2x2. La respuesta correcta es 3.

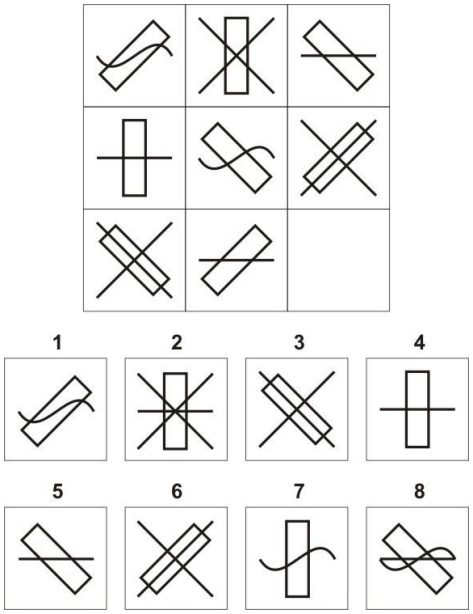


Figura 1.3. Problema con matriz de 3x3. La respuesta correcta es 7.

En cada versión, la dificultad de los problemas tiende a incrementarse conforme se avanza en la prueba, y el puntaje total obtenido se interpreta con respecto a los resultados típicos de las personas con la misma edad del participante. Por ejemplo, un puntaje de 32 es considerado alto para personas de 7 años, promedio para personas de 9.5 años, y bajo para personas de 16.5 años.

La prueba fue diseñada en 1936 por John Carlyle Raven con el objetivo de medir la habilidad de las personas para descubrir relaciones (Raven, J. C., 1936; Raven, 2011)[2]. Su naturaleza no verbal y la simplicidad de su aplicación permiten que la prueba se administre a personas de diferentes edades, niveles educativos y orígenes culturales, así como a personas con discapacidades motrices o mentales. Estas ventajas han hecho que la RPM se utilice en entornos educativos, profesionales, militares y de investigación alrededor del mundo.

En este proyecto la RPM es un elemento central. Por ello, en lo que resta de esta sección se expone una breve revisión de lo que la RPM representa en el campo de la psicología.

La teoría de la inteligencia de Spearman

John C. Raven diseñó la RPM para medir la *habilidad eductiva*, uno de los elementos definidos por Charles Spearman en su teoría de la inteligencia (Raven, 2011; Spearman, 1904).

Spearman notó que los resultados de varias pruebas de habilidad mental estaban correlacionados positivamente. Es decir, que las personas que obtenían buenos resultados en alguna de las pruebas tendían a obtener buenos resultados en las demás pruebas. Entonces propuso un modelo de dos factores para describir esta relación:

- Un factor general (*g*), que todas las pruebas medían en cierto grado, y que explicaba la correlación entre las pruebas,
- Un factor específico (*s*), que cada prueba medía de forma diferente, y que explicaba la variabilidad entre las pruebas.

Tiempo después propuso que *g* estaba formado por dos habilidades diferentes (Spearman, 1927):

- Una *habilidad eductiva*, que consiste en descubrir relaciones, o extraer significado a partir de información nueva,
- Una *habilidad reproductiva*, que consiste en aplicar información que se ha aprendido.

John C. Raven diseñó la RPM para medir la primera de estas habilidades, y la prueba Mill Hill de vocabulario para medir la segunda.

Spearman (1927) también observó que la correlación que había encontrado era menor en personas con *g* alto. Es decir, las personas con *g* alto tendían a obtener resultados más irregulares a lo largo de las diferentes pruebas, mientras que las personas con *g* bajo tendían a obtener resultados más uniformes. Este fenómeno es conocido como la "ley de Spearman de los rendimientos decrecientes" (SLDR en inglés) o la "hipótesis de la diferenciación", y ha sido confirmado posteriormente en grupos de adultos y niños utilizando diferentes conjuntos de pruebas cognitivas (Tucker-Drob, 2009).

Spearman aclaró que *g* no debía entenderse como una *inteligencia general*, sin embargo, es precisamente lo que se ha hecho en muchos trabajos posteriores (Raven, 2011). Spearman mismo mantiene diferentes puntos de vista con respecto a lo que significa *g*, desde una herramienta puramente estadística, hasta evidencia de algún proceso cognitivo común y subyacente a las pruebas. El debate sobre lo que *g* representa continúa en la actualidad.

[2] John C. Raven (citado como Raven, J. C.) fue el padre de John Raven (citado como Raven).

En la visión de Spearman toda persona es un genio en alguna actividad, y el problema es descubrir cuál es esta actividad (Spearman, 1924). Él afirmaba que las pruebas de las cuales había surgido *g* no debían utilizarse en las escuelas porque interferían con el verdadero propósito de la educación: descubrir los diferentes talentos de los estudiantes (Raven, 2011).

Relevancia de la prueba

Aunque la prueba de Raven fue originalmente diseñada para medir la habilidad eductiva, se ha descubierto que los resultados de esta prueba están altamente correlacionados con los resultados de varias pruebas de inteligencia, incluso cuando estas pruebas incluyen tanto problemas visuales como problemas verbales (Raven et al., 2004). Esto sugiere que la RPM está vinculada a habilidades cognitivas que van más allá de la habilidad eductiva y del razonamiento visual, como la habilidad de realizar analogías (Lovett et al., 2010). Lo anterior ha hecho que la RPM sea conocida como la mejor forma de medir el factor *g* mismo (Neisser, 1997).

Por otro lado, se han descubierto correlaciones entre un *g* alto y varias medidas convencionales de "éxito", como un mayor nivel de estudios, mayor productividad laboral y mayor nivel de ingreso. También se han encontrado correlaciones entre un *g* bajo y situaciones consideradas como desfavorables, como abandono de los estudios, embarazos no deseados, comportamiento criminal, pobreza e incluso un mayor riesgo de morir en un accidente de auto (Geary, 2005; Gottfredson, 1997, 2010; Neisser et al., 1996; O'Toole, 1990).

Así mismo, se han propuesto correlaciones entre *g* y diversas mediciones cerebrales, como el volumen total del cerebro, el espesor cortical (Figura 1.4), la tasa de metabolización de glucosa dentro del cerebro, y la velocidad de conducción nerviosa (Betjemann et al., 2009; Gottfredson, 2010; Jensen, 1998; Neisser et al., 1996; Reed et al., 1992; Shaw et al., 2006; van Leeuwen et al., 2009).

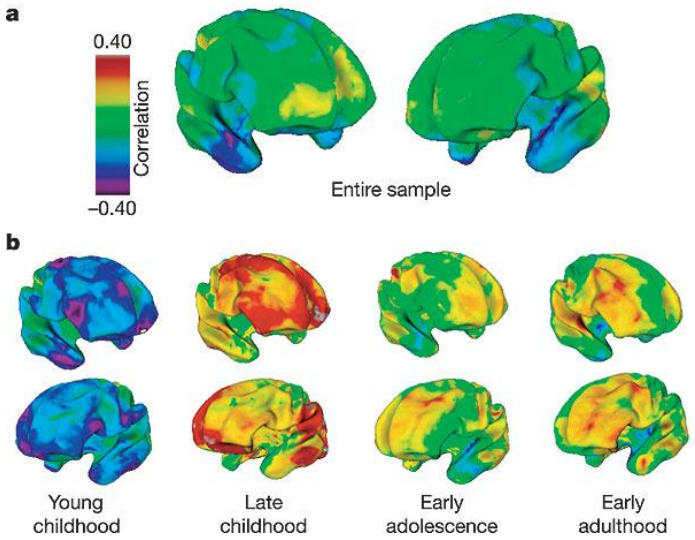


Figura 1.4. Correlación entre el IQ y el espesor cortical (Shaw et al., 2006). En este artículo se analizó a 307 personas y se utilizaron las pruebas Wechsler para medir el IQ. a) Correlación promedio de todas las personas analizadas. b) Correlación promedio por grupos de edad.

Las correlaciones mencionadas han contribuido a que el factor *g* sea considerado por muchos como un sinónimo de *inteligencia general* (Gottfredson, 1997; Raven, 2011), y a que la RPM sea vista como una *prueba de inteligencia*, o de *cociente intelectual* (IQ).

Estos puntos de vista cobran importancia porque la RPM y otras pruebas similares no son utilizadas sólo como instrumentos de investigación, sino también como herramientas de evaluación y selección en diversos entornos que incluyen desde escuelas y empresas hasta organizaciones civiles, gubernamentales y militares.

Por ejemplo, a principios del siglo veinte, varios lugares de Estados Unidos utilizaban pruebas de inteligencia para determinar si las personas tenían retraso mental, y si este era el caso, estas personas podían ser obligadas legalmente a esterilizarse (Dorr, 2008). En la actualidad, varios lugares de Estados Unidos utilizan pruebas de inteligencia para determinar si un criminal tiene retraso mental, en cuyo caso no se le puede aplicar la pena de muerte (McKinzey, 2003; Raven, 2005).

Por otro lado, el amplio uso de la RPM a lo largo de varias décadas ha permitido descubrir que los puntajes obtenidos en esta y otras pruebas de inteligencia han ido incrementándose de forma continua y aproximadamente lineal en muchas partes del mundo. Este fenómeno es conocido como el *efecto Flynn*, y aún no existe consenso cuáles son sus causas; tampoco se sabe con certeza si el incremento en puntajes indica un incremento real de la inteligencia (Flynn, 1999, 2007; Raven, 2000, 2002a).

Los datos acumulados mediante las pruebas de inteligencia también han revelado que el puntaje promedio de las personas de ascendencia negra es inferior al de las personas de ascendencia blanca (Neisser et al., 1996); las causas de esta diferencia de puntajes aún se desconocen, pero algunos han propuesto causas genéticas (Rushton et al., 2005), lo cual ha llevado a varios a afirmar que las pruebas de inteligencia han sido utilizadas como una herramienta de racismo científico (Flynn, 1999; Shönemann, 1997, 2005, 2009).

En suma, la forma en que se define y mide la inteligencia tiene profundas implicaciones en muchos ámbitos de la estructura social, y la prueba de Raven desempeña un papel importante en este tema.

Críticas a la prueba

Actualmente no existe consenso sobre cómo definir la inteligencia, ni sobre cómo ésta debe medirse (Flynn, 2007; Neisser et al., 1996; Raven, 2002b; Shönemann, 2005). Como se mencionó al introducir la teoría de Spearman, el factor *g* no fue originalmente pensado como un sinónimo de inteligencia general, y de hecho Spearman estaba en contra de esta idea. Él y otros investigadores han propuesto que el concepto de inteligencia debe tomar en cuenta la diversidad de las habilidades humanas en lugar de reducirse a un solo número (Flynn, 1999; Neisser et al., 1996; Raven, 2002b, 2011; Spearman, 1924). Sin embargo, Shönemann (2005) observa que si la inteligencia resulta ser multidimensional entonces no será posible "medirla", en el sentido de que no podrá asignarse un número real único a cada valor de inteligencia, ni podrán realizarse afirmaciones como "A es más inteligente que B".

Cuando se habló de la relevancia de la prueba, se explicó que la idea de *g* como una medida de inteligencia general proviene de una variedad de correlaciones (entre la RPM y otras pruebas, y entre *g* y varias mediciones sociales y biológicas). Sin embargo, algunos investigadores afirman que el factor *g* no es más que un artefacto estadístico, que la teoría de Spearman no lo define de forma única, y que las suposiciones que sustentan esta teoría no se cumplen en la práctica (Shönemann, 1997, 2005).

Por otro lado, algunos afirman que la metodología que se utiliza en la actualidad para medir la inteligencia es inadecuada, que los estándares técnicos son pobres, y que básicamente el campo de estudio se ha estancado y ha regresado a las raíces racistas que tenía en 1920 (Shönemann, 2005). John Raven, hijo del autor de la RPM, también es un crítico de los métodos actuales. Él propone que las pruebas de IQ deberían complementarse de modo que sea posible descubrir los diferentes talentos de las personas (Raven, 2002a), y afirma que no hacerlo puede tener graves consecuencias para la sociedad (Raven, 2002b).

Un reporte elaborado por la Asociación Estadounidense de Psicología (Neisser et al., 1996) concluye que el tema de la definición y medición de la inteligencia aún se encuentra lleno de incógnitas, así como polarizado a causa de sus implicaciones sociales y políticas. Quizás es una conclusión similar a la que propuso Spearman en 1927:

> "Los resultados de las pruebas y las tablas numéricas son acumulados; acciones que afectan el bienestar de las personas son propuestas, e incluso llevadas a cabo, con base en – ¡quién sabe qué!" (Spearman, 1927).

1.2 Trabajo previo

No se sabe con seguridad qué es lo que mide la RPM, ni si esta prueba representa una condición suficiente o necesaria para la inteligencia. Sin embargo, la prueba ha sido y es una herramienta muy útil para analizar los procesos involucrados en la inteligencia, ya sea natural o artificial. Es por ello que se han desarrollado varios programas de cómputo para resolver la RPM de forma automática, los cuales han permitido analizar las características de la RPM, estudiar la inteligencia humana y poner a prueba modelos computacionales de razonamiento. A la fecha se han desarrollado nueve programas de este tipo:

- Carpenter, Just y Shell (1990) (dos programas).
- Bringsjord y Schimanski (2003).
- Lovett, Forbus y Usher (2007, 2010) (dos programas).
- Cirillo y Ström (2010).
- Kunda, McGreggor y Goel (2010a, 2010b, 2011) (dos programas).
- Rasmussen y Eliasmith (2011).

Varios de estos programas buscan simular cómo las personas resuelven la prueba, por lo cual representan ejemplos de modelación cognitiva[3]. A continuación se describe brevemente cada uno de ellos, haciendo énfasis en las estrategias utilizadas, en los resultados obtenidos y en la relevancia de cada trabajo. Luego se presenta un análisis comparativo de los mismos.

[3] El objetivo de un modelo cognitivo es desarrollar una efectiva simulación de la solución un problema desde el punto de vista del humano (Laureano et al., 2000).

Carpenter et al. (1990)

Este trabajo aborda los procesos cognitivos involucrados en la RPM (en sus versiones estándar y avanzada) y los analiza en términos de cuáles son comunes en todas las personas y cuáles diferencian a las personas con alto puntaje de las personas con bajo puntaje. Para sustentar este análisis se aplicaron diversas pruebas cognitivas a estudiantes universitarios y se implementaron dos simulaciones computacionales: Fairaven y Betteraven. Cada simulación modeló el desempeño de los estudiantes con puntaje promedio y con puntaje alto en la prueba, respectivamente.

Análisis del problema

Tras analizar los problemas de la RPM, los autores propusieron cinco *tipos de reglas* para resolverlos:

- Constante en una fila: Una misma figura o atributo se repite a lo largo de cada fila de la matriz. Por ejemplo, en la Figura 1.5, la posición de los triángulos negros dentro de los cuadrados es constante en cada fila (arriba en la primera fila, abajo en la segunda fila, y arriba y abajo en la tercera fila). En la Figura 1.7, la orientación de la barra es constante en cada fila (vertical, horizontal y diagonal).
- Progresión cuantitativa por pares: En cada fila, un atributo se incrementa o decrementa entre cada par de celdas adyacentes. En la Figura 1.6, el número de cuadrados negros se incrementa entre cada par de celdas adyacentes en cada fila.
- Suma o resta de figuras: En cada fila, una figura es sumada a (o restada de) otra figura para producir una tercera figura. En la Figura 1.5, la celda izquierda de cada fila se suma con la celda central para obtener la celda derecha.
- Distribución de tres valores: Tres valores de un atributo están presentes en cada fila. En la Figura 1.7, el conjunto {rombo, cuadrado, triángulo} está presente en cada fila; también la textura de la barra sigue esta regla, pues el conjunto {negra, rayada, vacía} esta presente en las barras de cada fila.
- Distribución de dos valores: Similar al anterior, pero el tercer valor del atributo es nulo. En la Figura 1.8, diferentes líneas están presentes sólo en dos de las tres celdas de cada fila. En la Figura 1.7, si se eliminaran los tres triángulos de la matriz, el conjunto {rombo, cuadrado, nulo} estaría presente en cada fila y sería otro ejemplo de esta regla.

Estas reglas son aplicables a problemas con matrices de 2x2 o de 3x3, pero no a problemas con una sola imagen; estos últimos pueden resolverse mediante algoritmos perceptuales como los de continuación de líneas. Además, dos problemas de la RPM avanzada no pueden resolverse con las reglas presentadas porque involucran reglas de naturaleza diferente, como deformaciones o la interpretación de figuras como números (Figura 1.9). Algunos problemas de la RPM estándar también requieren este tipo de reglas.

Cada problema de la RPM puede involucrar una o varias de estas reglas, e incluso varias instancias de la misma regla. Por ejemplo, la Figura 1.7 presenta dos instancias de "distribución de tres valores" y una de "constante en una fila". Además, un mismo problema puede resolverse con diferentes conjuntos de reglas. Por ejemplo, la Figura 1.5 puede interpretarse como una "suma de figuras" o como una "distribución de tres valores", donde la posición de los triángulos en cada fila presenta los valores {izquierda, derecha, ambos}.

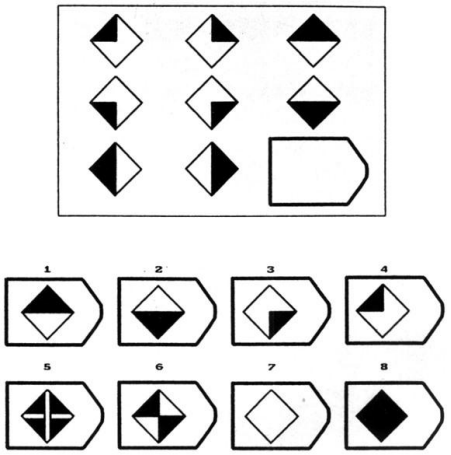


Figura 1.5. "Constante en una fila" y "suma de figuras" (Carpenter et al., 1990). La respuesta es 8.

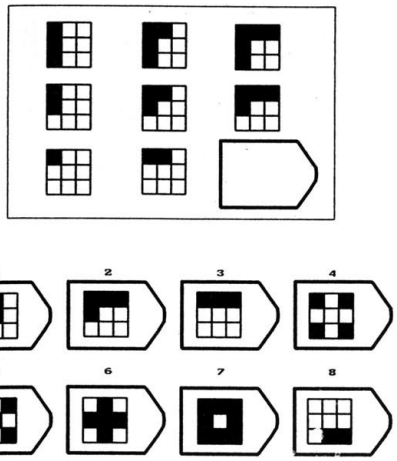


Figura 1.6. "Progresión cuantitativa por pares" (Carpenter et al., 1990). La respuesta es 3.

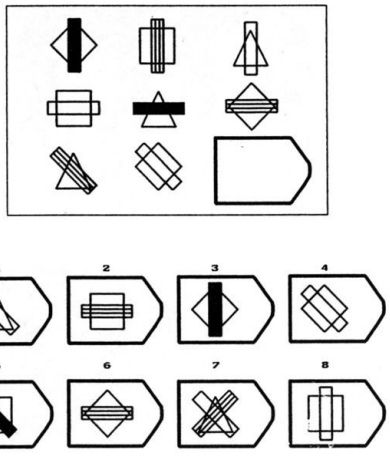


Figura 1.7. "Constante en una fila" y "distribución de tres valores" (Carpenter et al., 1990). La respuesta es 5.

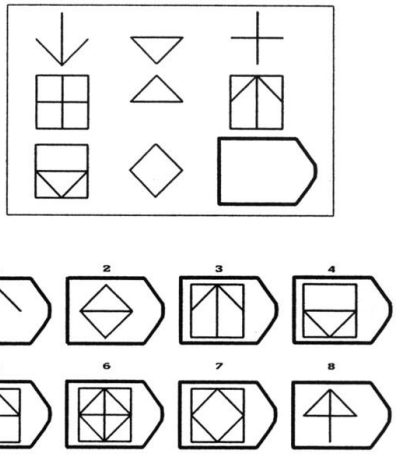


Figura 1.8. "Distribución de dos valores" (Carpenter et al., 1990). La respuesta es 5.

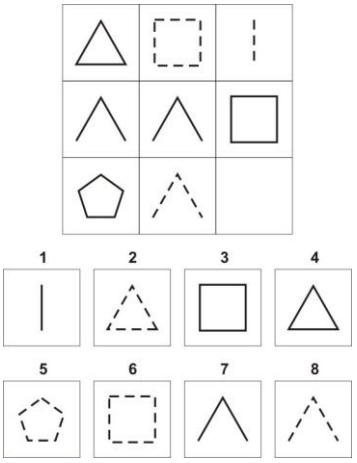


Figura 1.9. Interpretación de figuras como números. La respuesta es 4.

Las reglas siempre involucran filas de la matriz y no columnas, diagonales, ni otros patrones, ya que la mayoría de las personas analizadas en este trabajo analizaron las matrices por filas. Además, de acuerdo a Raven (2011), un requisito de las matrices progresivas es que la lógica detrás de cada matriz debe replicarse tanto en sus filas como en sus columnas, de manera que las personas que tomen la prueba puedan confirmar su razonamiento y estar completamente seguras de si han elegido la respuesta correcta. Por ello, un análisis por filas es suficiente para resolver los problemas de la RPM.

En los problemas con múltiples reglas, una de las dificultades principales es determinar cuáles figuras o atributos están gobernados por la misma regla, es decir, encontrar las *correspondencias* entre los elementos de la matriz. Por ejemplo, en la Figura 1.7 el participante debe descubrir que la orientación de las barras sigue una "constante en una fila", que la textura de las barras sigue una "distribución de tres valores" y que los elementos sobrantes (rombo, cuadrado y triángulo) siguen otra "distribución de tres valores". Los autores plantean la hipótesis de que el número de reglas en un problema afecta más a los procesos de administración de objetivos que a los procesos inductivos del participante[4].

Para analizar las estrategias que utilizan las personas al resolver la RPM se realizaron dos experimentos. En el primero se presentaron problemas de la RPM estándar y avanzada a 34 estudiantes universitarios, y mientras los resolvían se grabó el movimiento de sus ojos y los comentarios verbales que realizaban. Por razones técnicas, sólo se utilizaron 34 problemas de la RPM avanzada para analizar el movimiento ocular de los participantes. Este experimento permitió obtener las siguientes conclusiones:

- El número de reglas en un problema incrementa el tiempo requerido para resolverlo, e incrementa la probabilidad de resolverlo incorrectamente.
- Todos los participantes siguieron una estrategia incremental, pues las reglas que identificaban eran descritas de forma separada y con intervalos de tiempo relativamente largos entre las descripciones.
- Cada atributo fue descrito de forma separada, con intervalos de tiempo variables entre las descripciones.
- La inducción de cada regla requirió varias comparaciones entre parejas de celdas de la matriz, como se observó en las grabaciones de movimiento ocular.
- Se realizaron numerosas comparaciones entre parejas de celdas de la matriz, y entre celdas de la misma fila.

En el segundo experimento se administró la RPM avanzada a 45 estudiantes universitarios, siguiendo los procedimientos convencionales de la prueba. Además se presentó el problema de las Torres de Hanoi a los mismos estudiantes (Figura 1.10). Este problema consiste en mover una pila de discos desde un poste hasta otro obedeciendo las siguientes reglas:

- Sólo se puede mover un disco a la vez.
- Cada movimiento consiste en quitar el disco superior de uno de los postes y colocarlo en otro poste, encima de los discos que hubiera en ese poste.
- En ningún momento puede colocarse un disco encima de otro de menor tamaño.

[4] El razonamiento inductivo consiste en obtener conclusiones generales a partir de casos u observaciones particulares. Por ejemplo, si todos los cuervos que hemos visto son negros, tendemos a concluir que todos los cuervos son negros (aunque puede que no sea así). En este tipo de razonamiento las premisas apoyan en cierto grado a la conclusión, pero no implican que sea verdadera. Como se verá más adelante, en la RPM el participante debe inducir reglas observando las dos primeras filas de la matriz, y luego aplicarlas a la tercera fila para deducir la celda faltante.

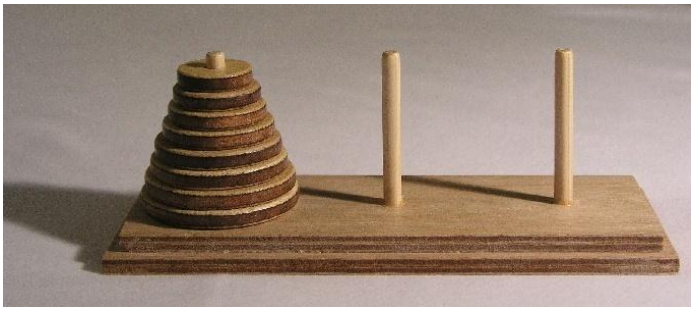


Figura 1.10. Problema de las Torres de Hanoi (Wikipedia).

A los participantes se les enseñó una estrategia recursiva para resolver este problema y se les pidió que lo resolvieran con 3 a 7 discos. Mediante este experimento se concluyó lo siguiente:

- Los resultados de la RPM avanzada y el desempeño en el problema de las Torres de Hanoi estuvieron altamente correlacionados.
- Ambas tareas pueden requerir procesos de abstracción similares, y una considerable habilidad para generar y administrar objetivos.
- A diferencia de la RPM, el experimento con las Torres de Hanoi requirió muy poco razonamiento inductivo por parte de los participantes, pues sólo debían aplicar la estrategia que se les había enseñado.
- La habilidad para generar y administrar objetivos tiene una influencia importante en el desempeño de las personas al resolver la RPM.

Las simulaciones

Con base en el análisis anterior se desarrollaron dos simulaciones computacionales. Ambas fueron implementadas como sistemas de producción usando CAPS (Concurrent, Activation-based Production System), a su vez implementado con OPS4.

En un sistema de producción, el conocimiento se representa en forma de reglas condición-acción (producciones). Si el contenido de la memoria de trabajo satisface las condiciones de una o más producciones, estas producciones son activadas y sus acciones son ejecutadas, modificando el contenido de la memoria de trabajo y activando así nuevas producciones.

CAPS difiere de otros sistemas de producción en varias características. Por ejemplo, en cada ciclo de ejecución se permite que todas las producciones activadas sean ejecutadas en paralelo, y las producciones pueden tener diferentes grados de activación.

Las simulaciones siguen una estrategia incremental, como se observó en los experimentos. Para cada problema, el funcionamiento general de ambas simulaciones es el siguiente:

- Las figuras y atributos de la primera fila de la matriz son procesados incrementalmente, y se identifica un conjunto de reglas (de entre los cinco tipos de reglas propuestos) que explique las variaciones entre estos elementos.
- Se repite lo anterior con las figuras y atributos de la segunda fila.
- Se encuentra un mapeo entre las reglas de la primera fila y las de la segunda fila.
- Las reglas de las primeras dos filas son generalizadas.
- Las figuras y atributos de la tercera fila son procesados incrementalmente.

- Las reglas generalizadas son aplicadas a la tercera fila para generar la celda faltante de la matriz.
- Se selecciona la respuesta más similar a la celda generada.

Las simulaciones no procesan imágenes directamente, sino que utilizan una representación simbólica de la RPM. Esta representación fue realizada manualmente, describiendo cada problema en términos de figuras básicas (como círculos, rectángulos y líneas) y sus atributos (como *textura rayada*). Los autores afirman que este hecho no compromete su análisis de las diferencias entre personas con puntajes altos y bajos por tres razones:

- La alta correlación entre la RPM y pruebas no visuales indica que los procesos de codificación visual no son una fuente importante de diferencias entre los individuos.
- Los experimentos realizados sugieren que las personas no tienen dificultad en reconocer las figuras de cada problema.
- Los experimentos indican que las personas sí tienen dificultad en encontrar las correspondencias entre las figuras y atributos de la matriz, y este proceso sí es modelado en las simulaciones.

La primera simulación, Fairaven, modela el desempeño de los estudiantes que obtuvieron un puntaje promedio en la RPM. La segunda, Betteraven, introduce varias mejoras a Fairaven para modelar el desempeño de los estudiantes con puntaje alto.

Fairaven

Esta simulación consta de 121 producciones y su estructura es la siguiente:

- Análisis perceptual (58 producciones): Analiza las figuras y atributos de la matriz.
    - Codificación: Se encarga de leer las figuras y atributos de cada problema desde un archivo hacia la memoria de trabajo. Estos elementos se leen de forma incremental, como se observó en los experimentos, y su lectura puede ser secuencial o guiada por las reglas que se ya han identificado en la matriz.
    - Correspondencias: Identifica a las figuras o atributos de cada fila que son gobernados por una misma regla, utilizando tres mecanismos:
        - Reglas identificadas: Trata de utilizar las reglas que se han identificado y generalizado en filas anteriores para definir las correspondencias de las demás filas.
        - Heurística de nombres: Plantea la hipótesis de que las figuras con nombres iguales se corresponden. Esta fue la estrategia más utilizada por los participantes de los experimentos.
        - Heurística de sobrantes: Supone que si tras aplicar los otros dos mecanismos sobra una figura o atributo en cada celda de una fila, entonces estos elementos sobrantes se corresponden.
    - Comparación por pares: Define las similitudes y diferencias entre los elementos que se han definido como correspondientes.
- Análisis conceptual (48 producciones): Identifica y generaliza reglas.
    - Identificación: Trata de explicar la variación entre los elementos de las primeras dos filas de la matriz, utilizando la información generada en la *comparación por pares* para identificar cuatro de los cinco tipos de reglas propuestos (Fairaven no incluye la regla de "distribución de dos valores").

    - Correspondencias: Encuentra un mapeo entre las reglas de la primera fila y las de la segunda fila.
    - Generalización: Combina las reglas correspondientes de las dos primeras filas para definir una forma general de estas reglas. Las reglas generalizadas utilizan variables en lugar de figuras y atributos específicos.
- Generación y selección de la respuesta (15 producciones): Utiliza las reglas generalizadas y el análisis perceptual de la tercera fila para generar una hipótesis de la celda faltante de la matriz, y luego selecciona la opción que resulte más similar a esta hipótesis. Esta estrategia es similar a la utilizada por los participantes con puntaje alto, quienes tratan de generar la celda faltante antes de examinar las posibles respuestas.

El análisis perceptual y el conceptual se realizan simultáneamente, ya que se identifican reglas conforme se analizan las figuras y atributos de la matriz.

Fairaven tiene cuatro limitaciones:

- No incluye la regla de "distribución de dos valores", debido a que sólo puede manejar reglas que involucran a las tres celdas de cada fila. Sin embargo, Fairaven trata de modelar a los participantes con puntaje promedio, los cuales con frecuencia no lograban resolver los problemas que incluían esta regla.
- Los tres mecanismos que incluye para identificar las correspondencias entre elementos no son suficientes para todos los problemas. En algunos casos asume correspondencias erróneas, lo cual lleva al programa a identificar reglas equivocadas.
- No tiene forma de corregir sus hipótesis (no es capaz de hacer *backtracking*).
- En problemas con tres o más reglas, el número de objetivos y el paralelismo de CAPS crean una carga de trabajo que el sistema no es capaz de manejar.

Betteraven

Esta simulación introduce las siguientes mejoras a Fairaven:

- Administrador de objetivos (15 producciones):
    - Este nuevo módulo permite ejercer un control más directo sobre todos los procesos de la simulación.
    - Se encarga de definir, monitorear y ajustar un árbol de objetivos, en el que el objetivo principal es resolver un problema de la RPM, y los subobjetivos incluyen analizar cada fila, comparar celdas adyacentes, encontrar correspondencias, etcétera.
    - Utilizando el árbol de objetivos, se encarga de asegurar que sólo un módulo de la simulación esté activo a la vez. En consecuencia, el análisis perceptual se realiza antes del conceptual, y no simultáneamente como en Fairaven.
    - Permite que las producciones del análisis perceptual se sigan ejecutando en paralelo (como en Fairaven), pero obliga a las del análisis conceptual a ejecutarse de forma serial (como se describe más abajo).
    - Permite que el sistema pueda modificar sus hipótesis cuando éstas no se cumplen (mediante *backtracking*). Estas hipótesis incluyen las reglas identificadas, los atributos considerados como relevantes y las correspondencias establecidas entre los elementos de la matriz.
- Análisis perceptual:

  - Se permite asignar un elemento nulo como correspondencia a otro elemento de la matriz. Esto permite manejar las reglas "distribución de dos valores" y "suma o resta de figuras", en las cuales un elemento de una celda puede no tener un elemento correspondiente en otra celda.
  - Se agrega una "heurística de atributos", que permite crear correspondencias entre elementos que tienen algún atributo en común (diferente del nombre de la figura, como la textura o la posición).
- Análisis conceptual:
  - Se incluye la regla de "distribución de dos valores".
  - La identificación de reglas ya no se realiza en paralelo, sino en el siguiente orden:
    1. Constante en una fila.
    2. Progresión cuantitativa por pares.
    3. Distribución de tres valores.
    4. Suma o resta de figuras.
    5. Distribución de dos valores.

    Este orden está basado en la simplicidad de las reglas y en reportes verbales de personas que tomaron la RPM. La identificación serial de las reglas permite realizar un procesamiento incremental de cada problema, similar al observado en los experimentos, y reduce la carga de trabajo del sistema.

Resultados y conclusiones

Como se mencionó al describir los experimentos, sólo se usaron 34 de los 48 problemas de la RPM avanzada al analizar el movimiento ocular de los participantes. Los autores limitaron la evaluación de las simulaciones a estos mismos 34 problemas. Además, como se comentó al describir los tipos de reglas propuestos, dos de estos problemas requieren reglas de naturaleza diferente, de modo que el máximo puntaje que podían obtener las simulaciones es 32. A continuación se presentan los resultados y conclusiones obtenidas por los autores.

Fairaven resolvió 23 problemas correctamente y Betteraven 32. Tras comparar los resultados de las simulaciones con los de los experimentos se obtuvieron las siguientes conclusiones:

- Fairaven y Betteraven modelaron correctamente el desempeño de los participantes con puntajes promedio y altos, respectivamente.
- Las simulaciones permitieron dividir a los problemas en grupos de creciente dificultad:
  - Los que sólo involucran a la regla "constante en una fila" son los más sencillos, pues sólo requieren recordar un valor a la vez (la constante de cada fila).
  - Los que sólo involucran "progresión cuantitativa por pares" requieren considerar dos figuras al mismo tiempo y realizar una generalización.
  - Los que involucran "distribución de tres valores" o "suma o resta de figuras" requieren considerar tres figuras al mismo tiempo.
  - Los que involucran varias instancias de reglas requieren más repeticiones del proceso de inducción, imponen una mayor carga en la memoria de trabajo y hacen más difícil el proceso de encontrar las correspondencias entre los elementos de la matriz. Este último proceso es una fuente importante de dificultad en la RPM.
- En cada problema, las reglas utilizadas por las simulaciones fueron consistentes con las utilizadas por los participantes.
- Las personas con puntaje alto tienen diferentes preferencias que las personas de puntaje promedio al inducir reglas, pues las primeras tienden a preferir reglas transformacionales

(como "suma o resta de figuras" o "progresión cuantitativa") en lugar de reglas distribucionales (como "distribución de dos (o tres) valores").
- Betteraven modeló correctamente la estrategia incremental de los participantes, pues induce una regla a la vez y cada regla inducida requiere varios ciclos de análisis perceptual y conceptual.

También se desarrollaron versiones "degradadas" de Betteraven y se concluyó lo siguiente:
- Al eliminar la regla "distribución de dos valores" su puntaje pasó de 32 a 23, igual al de Fairaven. Los autores concluyen que estos 9 problemas requieren establecer correspondencias abstractas (entre una figura o atributo y un elemento nulo) y un mejor manejo de objetivos.
- Al limitar el número de reglas que el programa podía mantener en la memoria de trabajo, antes de comenzar a eliminar objetivos de esta memoria, su puntaje se redujo en 11, 8 y 4 puntos para límites de 3, 4 y 5 reglas, respectivamente. Esto confirma la importancia de la habilidad para manejar múltiples objetivos al resolver la RPM.

Con respecto a las limitaciones de los modelos, se observó que:
- Las simulaciones no modelan los procesos de codificación visual de las personas, pues parten de una representación simbólica de la RPM. Sin embargo, los autores afirman que este hecho no compromete su análisis porque estos procesos no influyeron en el desempeño de las personas analizadas.
- A diferencia de las personas analizadas, las simulaciones no organizaron sus propios procesos tras leer las instrucciones de la prueba. Los autores argumentan que estos meta-procesos[5] tampoco son fuente de diferencias entre los individuos, pues todos los participantes comprendieron igualmente las instrucciones de la prueba. Además, afirman, estos meta-procesos podrían agregarse a las simulaciones sin modificar sus fundamentos.
- Las simulaciones no "inducen" reglas, pues sólo reconocen las reglas que les fueron programadas. Los autores afirman que el conocimiento de las reglas tampoco es fuente de diferencias, pues todas las personas fueron capaces de identificar los diferentes tipos de reglas (con excepción de "distribución de dos valores", que frecuentemente no era identificada por las personas con puntaje promedio).
- Los modelos están basados en experimentos realizados con estudiantes universitarios, quienes tienden a obtener puntajes mayores al promedio. Los autores afirman que los modelos propuestos podrían generalizarse para modelar a personas con puntajes más bajos.

En relación a las implicaciones de los modelos en el análisis de la inteligencia, se concluyó que:
- El razonamiento abstracto es un componente crucial de la inteligencia. Éste consiste en construir representaciones que dependen más de interpretaciones de alto nivel que de percepciones, y estas interpretaciones permiten la construcción de generalizaciones en el espacio y el tiempo. La mayoría de las teorías formales de la inteligencia incluyen al razonamiento abstracto. Además, la teoría del desarrollo cognitivo de Jean Piaget caracteriza el desarrollo intelectual como una progresión desde lo concreto hacia lo abstracto y simbólico.
- La resolución de la RPM requiere el uso de procesos de abstracción, y en las diferentes versiones de la RPM, los problemas más difíciles involucran reglas más abstractas. Las

[5] Procesos que organizan o controlan a otros procesos.

simulaciones permitieron identificar cuáles problemas y procesos involucrados en la RPM requieren abstracción, así como analizar cómo las personas difieren en su habilidad para razonar en diferentes niveles de abstracción al resolver la RPM.

- Un componente clave de la inteligencia analítica (o en la teoría de Spearman, la *habilidad eductiva*) es la habilidad para generar y administrar objetivos y sub-objetivos en la memoria de trabajo. La creación de sub-objetivos permite descomponer problemas complejos en simples, así como monitorear los progresos y fallos realizados en el proceso de solución.
- La administración de objetivos probablemente interactúa con otro factor determinante de la dificultad de los problemas: la novedad. Una tarea novedosa puede requerir la organización de objetivos, abstracción y reflexión, mientras que las tareas rutinarias requieren la aplicación de procedimientos aprendidos y pueden realizarse mediante comportamientos más basados en percepciones. Si las personas tuvieran práctica o adiestramiento en la RPM, la administración de objetivos en esta tarea se volvería rutinaria y la prueba sería más fácil.
- El análisis realizado con la RPM debería ser aplicable a otras tareas cognitivas complejas, debido a las correlaciones entre la RPM y otras pruebas y tareas de razonamiento, como el problema de las Torres de Hanoi. Varios investigadores sugieren que estas correlaciones indican la existencia de procesos cognitivos de alto nivel compartidos por estas tareas, y este trabajo ha ayudado a analizar estos procesos.
- Los modelos no sólo permitieron analizar los procesos que diferencian a las personas con puntajes altos y bajos, sino también los procesos que comparten ambos grupos, como la naturaleza incremental e iterativa del razonamiento, que involucra la descomposición de problemas en sub-problemas.

Los autores concluyen que la RPM es un test de inteligencia que mide:

- La capacidad común para descomponer problemas en elementos manejables e iterar sobre estos elementos.
- La capacidad diferenciada para administrar jerarquías de objetivos generadas por la descomposición de los problemas.
- La capacidad diferenciada para generar abstracciones de alto nivel.

Bringsjord et al. (2003)

Este trabajo menciona el uso de un demostrador de teoremas para resolver problemas selectos de la RPM representados en lógica de primer orden. Sin embargo, los autores comentan que el trabajo se desarrolló bajo contrato de una empresa privada y no describen los detalles de su implementación ni sus resultados.

En este artículo también se propone una definición de *inteligencia artificial* basada en pruebas psicométricas:

> "La *inteligencia artificial psicométrica* es el campo dedicado a construir entidades que procesen información y sean capaces de alcanzar un desempeño sólido en todas las pruebas de inteligencia y habilidad mental establecidas y validadas, no sólo incluyendo a las restrictivas pruebas de IQ, sino también a las pruebas de creatividad artística y literaria, de habilidad motriz, etcétera".

Los autores afirman que esta definición extiende la idea de la prueba de Turing como un método para determinar si una entidad es inteligente, con la ventaja adicional de que permite dividir este problema en varios subproblemas que pueden abordarse por separado y con diferentes técnicas. De acuerdo a esta definición, los programas desarrollados para resolver la RPM representan avances en el campo de la inteligencia artificial psicométrica.

Lovett et al. (2007, 2010)

Este trabajo presenta un modelo para resolver la RPM estándar basado en dos componentes:

- Una representación espacial cualitativa y jerárquica.
- Un modelo de analogía basado en el mapeo de estructuras.

El modelo fue implementado en 2007 y en 2010. En ambos casos, los programas no procesan imágenes directamente, sino diagramas dibujados manualmente en PowerPoint, en los que cada problema de la RPM se representa mediante figuras básicas como líneas y polígonos.

Implementación 2007

En esta implementación se trató de resolver las secciones B y C de la RPM estándar sin utilizar procesos diseñados específicamente para esta tarea. Para el primer componente del modelo (la representación espacial cualitativa) se utilizó el sistema sKEA (Sketching Knowledge Entry Associate), el "primer sistema de dominio general para la comprensión de dibujos". Este sistema tiene las siguientes características:

- Está diseñado para ser de dominio general, y varios de sus componentes han sido utilizados en diferentes tareas de razonamiento espacial.
- Utiliza representaciones espaciales inspiradas en estudios de la percepción humana.
- No procesa imágenes directamente, sino diagramas realizados a mano en PowerPoint, utilizando figuras básicas como líneas y polígonos. PowerPoint representa estas figuras mediante el conjunto de instrucciones de dibujo definido en el formato WMF (Windows Metafile Format).
- Permite etiquetar manualmente cada figura usando un conjunto de categorías conceptuales.
- Trata a las líneas que forman una figura cerrada como una sola entidad, a la que se le puede asignar un color de relleno y un color de borde. Las líneas que no forman figuras cerradas se tratan como entidades independientes.
- Realiza una identificación rudimentaria de texturas, tratando como una sola entidad a los conjuntos de líneas paralelas que no forman parte de figuras cerradas; además, determina si alguna figura cerrada envuelve a estas líneas, de modo que la textura pueda asociarse a esta figura.
- Calcula automáticamente varias relaciones espaciales cualitativas entre las entidades de un diagrama, como "a la izquierda de", "arriba de" y "dentro de".
- Construye una representación cualitativa y jerárquica de un diagrama, utilizando las etiquetas conceptuales agregadas manualmente, y las relaciones espaciales cualitativas calculadas automáticamente.

Para el segundo componente del modelo (el modelo de analogía) se utilizó el sistema SME (Structure Mapping Engine), el cual tiene las siguientes características[6]:

- Es un modelo de analogía de dominio general, fundamentado en la teoría de la analogía de Gentner (1983).
- Está basado en el concepto de *mapeo de estructuras*, el cual define a la analogía y a la similitud en términos de un proceso de comparación. Este proceso opera sobre representaciones estructuradas, es decir, descripciones simbólicas que involucran entidades, atributos y relaciones (incluyendo relaciones entre relaciones, o *relaciones de orden superior*).
- Dada una descripción base y una descripción objetivo, el sistema calcula *mapeos* que constan de:
    - Correspondencias, que indican cómo se alinean los elementos de ambas descripciones.
    - Inferencias candidatas, que están basadas en las diferencias entre los elementos correspondientes.
    - Un puntaje de similitud, que indica la calidad del mapeo y que está basado en una evaluación estructural, es decir, en qué tanto y a qué profundidad se alinearon las estructuras de ambas descripciones.
- Permite definir restricciones en los mapeos, como requerir que ciertas correspondencias estén presentes (o ausentes), o requerir que las correspondencias sólo involucren ciertos tipos de elementos.

Usando estos dos componentes, los problemas de la RPM se resuelven con una variante del "mapeo de dos etapas", proceso propuesto por Tomai et al. (2005) para resolver problemas de analogía geométrica. Para problemas con matriz de 2x2 se sigue el siguiente procedimiento:

1. Se dibuja el problema manualmente en PowerPoint y se introduce al sistema sKEA.
2. sKEA construye una representación cualitativa y jerárquica de cada celda de la matriz y de cada posible respuesta.
3. SME analiza a la matriz por filas:
    - Construye un mapeo entre las dos celdas de la primera fila, y define un conjunto de diferencias y similitudes entre estas celdas (conjunto base). En este proceso se identifican transformaciones entre figuras, como rotaciones, escalamientos y reflexiones.
    - Repite el paso anterior con la celda de la segunda fila y cada posible respuesta, es decir, se definen 6 conjuntos adicionales de diferencias y similitudes (conjuntos objetivo).
    - Mapea y compara al conjunto base con cada conjunto objetivo, lo cual permite asignar un puntaje de similitud a cada posible respuesta.
4. SME analiza a la matriz por columnas (de forma análoga al paso 3), de modo que se asigna un segundo puntaje de similitud a cada posible respuesta.
5. Se promedian los dos puntajes de similitud de cada respuesta, y se selecciona la respuesta con el mayor puntaje.

La idea del procedimiento anterior es dividir a la matriz de 2x2 en dos problemas de analogía. Si cada celda se etiqueta con una letra, como en la Figura 1.11a, estos dos problemas de analogía son: AB→CX (A es a B, como C es a X) y AC→BX.

[6] Los detalles del sistema SME pueden consultarse en Falkenhainer (1989).

Figura 1.11. Etiquetas para las celdas de la matriz. a) Matriz de 2x2. b) Matriz de 3x3.

Donde X se sustituye con cada una de las posibles respuestas. Para resolver problemas con matriz de 3x3 se sigue un procedimiento similar, pero se divide a la matriz en cuatro problemas de analogía. Usando las etiquetas de la Figura 1.11b, estos problemas son: EF→HX, EH→FX, GH→HX y CF→FX. De este modo, cada respuesta recibe cuatro puntajes de similitud. Sin embargo, en este caso el puntaje final se calcula como el máximo de los cuatro puntajes, y no como el promedio.

Esta implementación se evaluó con las secciones B y C de la RPM estándar, y logró resolver 12 y 10 problemas de cada sección, respectivamente. Los autores observan que los 2 problemas de la sección C que el sistema no logró resolver no son los más difíciles de esta sección de acuerdo a estándares humanos. Estos dos errores se deben a las limitaciones del sistema para manejar texturas y para manejar figuras cuyo número de elementos cambia de una celda a otra.

Los autores obtienen dos conclusiones de esta implementación:

- La representación espacial cualitativa y jerárquica propuesta codifica suficiente información para resolver 22 de 24 problemas de la RPM estándar. Lo anterior, sumado a varios estudios de la percepción humana, sugiere que esta representación captura propiedades importantes de la representación visual humana.
- El sistema SME, combinado con el "mapeo de dos etapas", fue capaz de resolver problemas de la RPM estándar con dificultad baja a intermedia, sin utilizar procesos diseñados especialmente para esta tarea. Esto apoya la idea de que SME modela procesos cognitivos generales de mapeo estructural.

Implementación 2010

Esta implementación está basada en los mismos componentes que la anterior (la representación espacial cualitativa y el modelo de analogía). Sin embargo, en este caso se utilizó el sistema CogSketch para la representación espacial. Este sistema introduce los siguientes cambios al sistema sKEA:

- Realiza un mejor manejo de estructuras jerárquicas, pues procesa las representaciones espaciales en tres niveles: bordes, objetos y grupos.
- Crea grupos de objetos con base en su similitud y proximidad. Entre los grupos creados están:
    - Clases de forma estricta: Grupos de objetos que son idénticos.
    - Clases de forma equivalente: Grupos de objetos que son transformaciones de un mismo objeto (por ejemplo, triángulos equiláteros con diferente orientación y tamaño).
- Es capaz de segmentar objetos con base en comparaciones (Figura 1.12).
- Calcula más relaciones cualitativas entre entidades, como la relación de "intercepta a".
- Calcula más atributos cualitativos de las entidades, como su grado de simetría o la convexidad de sus esquinas.

- Además de las transformaciones manejadas por sKEA (rotaciones, escalamientos y reflexiones), CogSketch es capaz de manejar *deformaciones*, como escalamientos irregulares o la suma o resta de partes.

Figura 1.12. Segmentación basada en comparaciones. Al comparar la figura de la izquierda con la del centro, la de la izquierda es segmentada en dos objetos, como muestra la figura derecha.

El procedimiento para resolver los problemas de la RPM también es diferente, pues en esta implementación se sigue más de cerca el modelo propuesto por Carpenter et al. (1990). El procedimiento para resolver problemas con matriz de 3x3 es el siguiente:

1. Se dibuja el problema manualmente en PowerPoint y se introduce a CogSketch. En esta implementación, la segmentación manual de las imágenes se realizó siguiendo los principios de agrupación Gestalt (Palmer et al., 1994).
2. CogSketch construye una representación cualitativa y jerárquica de cada celda de la matriz y de cada posible respuesta, definiendo bordes, objetos y grupos.
3. Se utiliza SME para establecer mapeos entre las celdas de la primera fila de la matriz. Con base en estos mapeos, se construye un *patrón de variación*: una representación de cómo cambian los objetos a lo largo de esta fila.
4. Se repite el paso 3 con la segunda fila de la matriz.
5. Se utiliza SME para comparar los patrones de las primeras dos filas y evaluar su similitud.
6. Si los patrones no son suficientemente similares, se cambia la *estrategia* del modelo y se regresa al paso 3. De lo contrario se procede al paso 7.
7. Se construye una generalización que representa los elementos comunes de los dos patrones.
8. Se agrega cada posible respuesta a la tercera fila, y se usa SME para construir un patrón de variación en cada caso (las matrices de 3x3 tienen ocho posibles respuestas, de modo que se construyen 8 patrones de variación adicionales).
9. Se usa SME para comparar cada patrón de variación de la tercera fila con la generalización construida en el paso 7.
10. Se selecciona la respuesta que produjo el patrón de variación más similar a la generalización. En caso de empates, no se selecciona ninguna respuesta.

Para generar los patrones de variación, el sistema utiliza cuatro *estrategias*. Los autores afirman que, a diferencia de los tipos de reglas propuestos por Carpenter et al. (1990), estas estrategias no son específicas para resolver la RPM y podrían aplicarse a otros problemas espaciales. Las estrategias son:

- Diferencias: Se analizan los cambios que sufren los objetos a lo largo de una fila, y estos cambios se representan mediante las siguientes cinco expresiones:
    - Identidad: El objeto no cambia.
    - Transformación: El objeto sufre una rotación, escalamiento o reflexión.
    - Deformación: El objeto sufre un escalamiento irregular, o pierde o gana partes.
    - Sustitución de forma: La forma del objeto es sustituida por otra (por ejemplo, de cuadrado a círculo).
    - Suma o resta: Un objeto es agregado o eliminado.

Estas expresiones se complementan con los cambios adicionales que existan entre los objetos, como cambios de color o de posición relativa.

- Diferencias avanzadas: Es igual a la estrategia anterior, pero se agregan restricciones a SME para que sólo se permitan mapeos entre objetos idénticos. Esto evita que el sistema genere las expresiones "transformación", "deformación" y "sustitución de forma", pero facilita la generación de la expresión "suma o resta", lo cual es necesario en ciertos problemas.
- Literal: Se analiza lo que está presente en las celdas de una fila, en lugar de los cambios entre las celdas. Esta estrategia es similar a la regla "distribución de tres valores" definida por Carpenter et al. (1990), pues busca identificar los objetos que están presentes en la fila sin analizar los cambios entre ellos.
- Literal avanzada: Es igual a la anterior, pero los objetos de cada celda son manejados de forma diferente:
    - Si cada celda de la fila sólo contiene un objeto, entonces cada uno de los atributos del objeto se maneja como una entidad independiente. Esto permite resolver problemas como el de la Figura 1.13.
    - Si cada celda contiene varios objetos, entonces cada objeto se maneja de forma independiente, eliminando las relaciones espaciales entre los objetos, y eliminando la asociación entre los objetos y las celdas a las que pertenecen. Esto permite resolver problemas como el de la Figura 1.14.

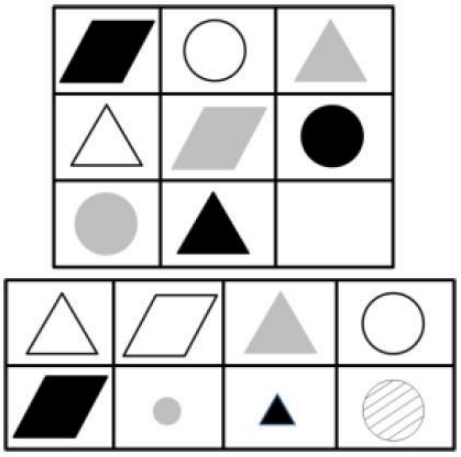


Figura 1.13. Las figuras y los colores de relleno siguen patrones diferentes, de modo que deben tratarse por separado (Lovett et al., 2010). La respuesta correcta es el cuadrilátero sin relleno.

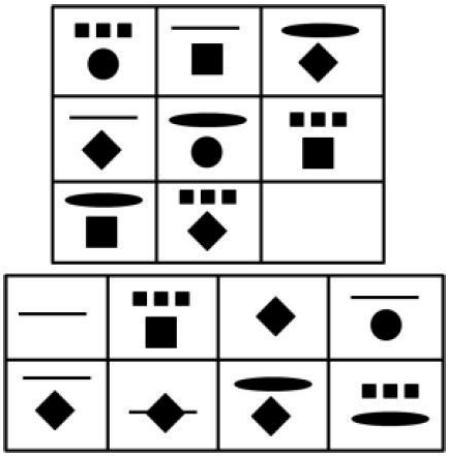


Figura 1.14. En cada fila, las figuras de arriba siguen un patrón diferente que el de las de abajo, por lo que deben tratarse por separado (Lovett et al., 2010). La respuesta correcta es el círculo con una línea encima.

El sistema sólo cambia de estrategia cuando los patrones de las primeras dos filas no resultaron suficientemente similares, y las estrategias son utilizadas en el siguiente orden: diferencias, literal, literal avanzada y diferencias avanzadas. Si los patrones no cumplen el criterio de similitud con ninguna de las estrategias, entonces el sistema continúa con el paso 7 del procedimiento usando la estrategia "diferencias" o "diferencias avanzadas" (la que haya producido la mayor similitud entre los patrones).

Para resolver problemas con matriz de 2x2 se sigue un procedimiento similar, pero sólo se usa la estrategia "diferencias". Como en el procedimiento anterior, se construye un patrón de variación

para la primera fila y un patrón de variación para cada posible respuesta. Luego se compara el patrón de la primera fila con el patrón de cada respuesta, y se elige la respuesta con el patrón más similar. Si ninguna respuesta produjo un patrón suficientemente similar, se repite este procedimiento analizando a la matriz por columnas.

Esta implementación se evaluó con las secciones B a E de la RPM estándar. La sección A no se utilizó porque involucra más habilidades perceptuales que razonamiento por analogía. La sección B utiliza matrices de 2x2 y las demás utilizan matrices de 3x3. El programa resolvió correctamente 44 de los 48 problemas. Los 4 problemas incorrectos son de los más difíciles de acuerdo a los estándares humanos, y el sistema no logró resolverlos debido a que no representa suficientes atributos de los objetos y grupos.

Los autores presentan las siguientes conclusiones:

- Usando el modelo propuesto fue posible simular el desempeño típico de las personas en la RPM.
- El mapeo de estructuras (SME) tuvo un papel importante en el modelo, pues fue utilizado para realizar muchas de las comparaciones involucradas.

Como trabajo futuro proponen utilizar el modelo para investigar:

- Cómo las personas realizan representaciones espaciales, cómo resuelven problemas espaciales y cómo realizan comparaciones por analogía.
- Qué hace a un problema más difícil que otro.
- Qué hace que una persona tenga un mejor desempeño que otra al resolver problemas espaciales.

Cirillo et al. (2010)

En este trabajo se implementó un programa para resolver las secciones C, D y E de la RPM estándar (éstas sólo contienen matrices de 3x3). Este programa tiene las siguientes características:

- No procesa imágenes directamente, sino representaciones vectoriales (simbólicas) de cada problema, elaboradas manualmente usando el formato XAML (eXtensible Application Markup Language) y similares a las utilizadas por Lovett et al. (2007).
- Utiliza una representación espacial jerárquica que permite el análisis de cada problema en varios niveles de organización.
- Utiliza un modelo cognitivo simple basado en patrones, similares a los tipos de reglas propuestos por Carpenter et al. (1990).
- Genera a la celda faltante de cada matriz, total o parcialmente, sin tomar en cuenta a las respuestas incluidas en el problema.

Representación espacial jerárquica

De acuerdo a la Teoría de la Información Estructural (Leeuwenberg, 1971), la mejor interpretación de un estímulo es la que produce la representación más simple para el estímulo. Además, las representaciones más simples implican la organización jerárquica de los estímulos en términos de relaciones todo-parte (van der Helm, 2007). Por ello, cada problema de la RPM es interpretado mediante varios niveles de organización.

La primera etapa del programa lee un problema en formato XAML y lo representa mediante un grafo en capas (Figura 1.15).

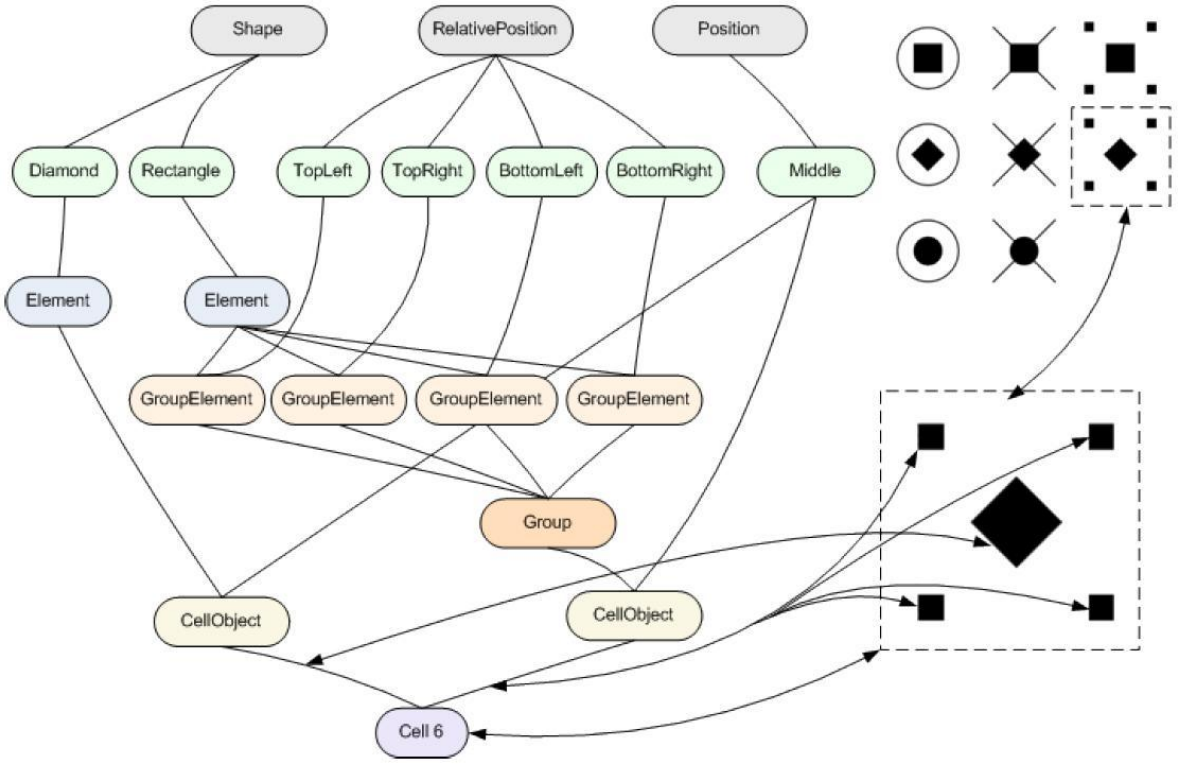


Figura 1.15. Representación jerárquica de una celda (Cirillo et al., 2010). Algunos atributos se omitieron por claridad.

Cada capa del grafo representa un nivel de organización. Las aristas del grafo sólo conectan nodos en capas diferentes, y el modelo consta de ocho capas:

- A*tributos*: Representa atributos relevantes de las figuras, extraídos directamente desde la representación XAML (como *forma* o *grosor de línea*) o calculados a partir de ésta (como *caja delimitadora* o *posición del centro*).
- *Valores de atributos*: Representa valores de los atributos (como el valor *rectángulo* para el atributo *forma*).
- *Modelos de elementos*: Agrupa los *valores de atributos* que son utilizados juntos en varias figuras de la matriz.
- *Instancias de elementos*: Representa figuras sin una posición definida.
- *Posiciones relativas*: Representa figuras con una posición relativa, la cual es definida con respecto a otras figuras de su mismo grupo.
- *Grupos*: Representa grupos de figuras, y permite realizar comparaciones entre figuras y grupos de figuras.
- *Posiciones absolutas*: Representa figuras o grupos con una posición absoluta, la cual es definida con respecto a una celda de la matriz.
- *Celdas*: Representa a cada celda de la matriz.

Nótese que la capa *posiciones absolutas* no representa valores de posición (como la capa *valores de atributos*), sino figuras o grupos que tienen una posición absoluta definida. Algunos problemas no

requieren la identificación de grupos o modelos, por lo que las capas *modelos de elementos*, *posiciones relativas* y *grupos* no siempre se utilizan. Por ejemplo, en la Figura 1.15, la celda resaltada de la matriz se representa de la siguiente forma:

- Los tres nodos de la primera fila representan *atributos*.
- Los siete nodos de la segunda fila representan *valores de atributo*.
- El nodo *Element* (izquierda) representa una figura sin posición cuyo atributo *forma* es *diamante* (o rombo).
- El nodo *Element* (derecha) representa una figura sin posición cuyo atributo *forma* es *rectángulo* (los cuadrados son un tipo de rectángulo).
- Cada nodo *GroupElement* representa un cuadrado con cierta posición relativa con respecto a los demás de su grupo, y está conectado con el nodo *Element* (derecha) y un nodo de *valor de atributo* para el atributo *posición relativa*.
- El nodo *Group* representa un grupo de cuatro cuadrados con posiciones relativas, pero sin posiciones absolutas.
- El nodo *CellObject* (izquierda) representa un diamante con cierta posición absoluta, y está conectado con el nodo *Element* (izquierda) y el nodo *Middle*.
- El nodo *CellObject* (derecha) representa un grupo de cuatro cuadrados con cierta posición absoluta, y está conectado con el nodo *Group* y el nodo *Middle*.
- El nodo *Cell 6* representa a la celda.

La Figura 1.16 muestra la representación de un problema completo. En este caso no se utilizaron grupos ni posiciones relativas. Como se mencionó, el programa no toma en cuenta a las respuestas incluidas en el problema.

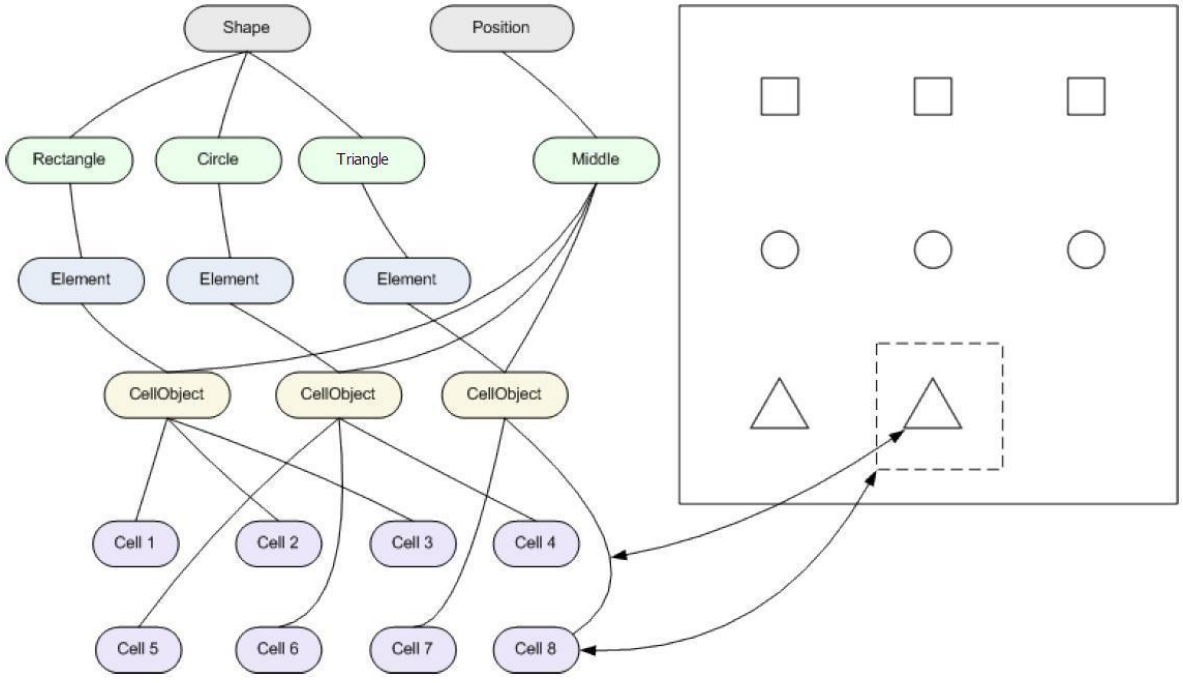


Figura 1.16. Representación jerárquica de un problema (Cirillo et al., 2010). Algunos atributos se omitieron por claridad.

El grafo se construye a partir de la representación XAML de cada problema, realizando la siguiente secuencia de etapas:

1. Inicialización: Se lee el archivo XAML y se definen los *atributos*, *valores de atributos*, *instancias de elementos*, *posiciones absolutas* (CellObjects) y *celdas*.
2. Extracción de atributos comunes: Los *valores de atributos* que están presentes en todas las *instancias de elementos* se eliminan del grafo; posteriormente se agregarán directamente a la solución generada.
3. Organización de instancias de elementos: Se eliminan las *instancias de elementos* y *posiciones absolutas* repetidas, pues no es necesario conservar varios nodos idénticos en el grafo.
4. Creación de modelos de elementos: Se identifican los *valores de atributos* que aparecen juntos en varias *instancias de elementos*; esto permite inferir la presencia de *valores de atributos* adicionales a partir de la presencia de uno de ellos.
5. Extracción de elementos comunes: Se eliminan las *posiciones absolutas* (CellObjects) que aparecen en todas las celdas de la matriz; posteriormente se agregarán directamente a la solución generada.
6. Creación y organización de grupos: Se agrupan las *instancias de elementos* que aparecen juntas en varias celdas de la matriz.

Esta representación permite describir los problemas de la RPM de una forma clara y económica, y permite abstraer los elementos gráficos de la lógica detrás de ellos.

Modelo cognitivo

Los autores proponen que, en los problemas de la RPM, la mejor respuesta es la que permite describir a la matriz de la forma más simple, es decir, minimiza la complejidad de la matriz. La complejidad de Kolmogorov proporciona una definición formal de la complejidad, pero ésta resulta incomputable (no puede calcularse usando tiempo y espacio finitos). Por otro lado, la Teoría de la Simplicidad (Chater et al., 2003) propone que, cuando se trata de sistemas cognitivos, la complejidad es relativa al observador y debe definirse mediante un modelo cognitivo. Por lo anterior, y para aprovechar las estrategias utilizadas por las personas al resolver la RPM, se propone un modelo cognitivo simple con las siguientes características:

- Cada matriz se procesa por filas o por columnas, identificando patrones que se cumplan en las dos primeras filas (o columnas) y aplicándolos a la tercera fila (o columna).
- Usando la representación jerárquica descrita, los problemas se analizan en cinco niveles de abstracción, cada uno con mayor detalle que el anterior:
    - Celdas completas (capa *celdas*).
    - Conteo de los objetos en cada celda.
    - Objetos con una posición absoluta definida (capa *posiciones absolutas*).
    - Objetos sin una posición definida (capa *instancias de elementos*).
    - Modelos de elementos (capa *modelos de elementos*).
- Sólo se pasa a un nivel más bajo cuando no fue posible generar la solución en los niveles anteriores.
- El modelo es capaz de identificar siete *patrones* diferentes:
    - Identidad: Una o más entidades son idénticas a lo largo de una fila (o columna). Este patrón puede aplicarse en cualquiera de los cinco niveles de abstracción, de modo que permite predecir celdas completas, conteos de objetos, etcétera. Además puede aplicarse en matrices de 2x2, 3x3 u otros tamaños.
    - Distribución de entidades: Varias entidades se repiten en todas las filas (o columnas) pero en diferente orden. Como el anterior, este patrón también puede aplicarse en cualquier nivel de abstracción y en matrices de cualquier tamaño.

- Progresión numérica: A lo largo de una fila (o columna), el número de objetos en cada celda aumenta o decrece de forma constante entre celdas adyacentes. Esta constante se identifica comparando las dos primeras celdas, y se verifica en las demás. Por tanto, este patrón sólo puede aplicarse en matrices de 3x3 o mayores, y en el nivel de conteo de objetos. Además, para evitar detectar patrones erróneos, se verifica que la progresión se cumpla en toda la matriz y no sólo en algunas de sus filas (o columnas).
- Traslación: A lo largo de una fila (o columna), las entidades se desplazan de forma constante entre celdas adyacentes. Este patrón sólo se aplica en el nivel de objetos con posición absoluta definida.
- Función AND: La última celda de una fila (o columna) contiene exactamente los elementos comunes a todas las demás celdas de la fila (o columna). Este patrón sólo se aplica en el nivel de celdas completas, y en matrices de 3x3 o mayores.
- Función OR: Es igual al patrón anterior, pero la última celda contiene la suma de los elementos de todas las demás celdas de su fila o columna.
- Función XOR: Igual al anterior, pero la última celda contiene los elementos que aparecen exactamente una vez en todas las demás celdas de su fila o columna.

Estos patrones se definieron con base en los tipos de reglas descritos por Carpenter et al. (1990), y mediante introspección. Para resolver un problema se prueban todos los patrones en cada nivel de abstracción, hasta que se genere la celda faltante en el nivel de celdas completas, o hasta que el número de objetos generados en la celda faltante sea igual al que se infirió en el nivel de conteo de objetos. Los patrones y los niveles de abstracción se prueban en el orden listado arriba.

Resultados y conclusiones

El modelo se implementó en el lenguaje C#.Net 3.5 y logró resolver 8, 10 y 10 problemas en las secciones C, D y E de la RPM estándar, respectivamente. Cada sección contiene 12 problemas, de modo que el sistema resolvió 28 de 36 problemas.

De los 8 problemas no resueltos, en 4 el sistema generó soluciones parciales y en 4 no generó ninguna solución. Algunos de estos problemas involucran patrones no programados en el sistema, como deformaciones o la interpretación de figuras como números (Figura 1.9). Otros requieren considerar a las posibles respuestas, ya que la solución no puede generarse por completo usando sólo la información incluida en la matriz; en estos casos, la información faltante concierne la posición, tamaño u orientación exacta de los elementos en la solución.

Con respecto al programa, los autores presentan las siguientes conclusiones:

- Algunos de los patrones programados se utilizaron más veces que otros al resolver la prueba, y de hecho es posible resolver muchos problemas usando pocos patrones.
- Al cambiar el orden de los patrones se resolvieron los mismos problemas, pero varios de ellos se resolvieron con diferentes patrones o en diferentes niveles de abstracción. Por tanto, el orden de los patrones sí influyó en el comportamiento del sistema.
- Los problemas se resolvieron usando los diferentes niveles de abstracción definidos, de modo que estos niveles resultaron adecuados para reconocer los patrones especificados.
- La modularidad del programa permite agregar nuevos patrones de una forma conveniente.
- El programa representa un ejemplo de *inteligencia artificial antropomórfica*, pues varios de sus elementos están inspirados en la cognición humana:

    - Utiliza una representación jerárquica inspirada en estudios de la percepción visual humana.
    - Utiliza un modelo cognitivo, aunque éste es rudimentario.
    - Utiliza patrones que han descrito las personas al resolver la RPM.
    - En cada problema, genera la solución sin tomar en cuenta a las posibles respuestas, una estrategia observada por Carpenter et al. (1990) en las personas que obtienen puntajes altos en la prueba.
- El programa no procesa imágenes directamente, sino representaciones vectoriales. Sin embargo, probablemente la RPM no pretende medir la capacidad de las personas para identificar figuras. Además, el programa puede extenderse para manejar imágenes sin procesar.

En relación a la RPM, los autores concluyen lo siguiente:

- Algunos problemas no están suficientemente especificados, lo cual hace imposible resolverlos de forma precisa sin considerar a las posibles respuestas.
- Aunque varios de los patrones utilizados se repiten a lo largo de la prueba, algunos aparecen sólo en un problema, lo cual hace difícil definir un modelo general para resolver la RPM.

Kunda et al. (2010a, 2010b, 2011)

Los trabajos anteriores utilizaron representaciones simbólicas para resolver la RPM. Sin embargo, Hunt (1974) propuso que existen dos estrategias para resolver la prueba:

- Gestalt, que utiliza representaciones visuales y operaciones perceptuales como la continuación y superposición de figuras.
- Analítica, que utiliza representaciones proposicionales y operaciones lógicas.

Varios análisis de la RPM estándar clasifican a los problemas en dos categorías: los que pueden resolverse mediante operaciones Gestalt y los que pueden resolverse mediante razonamiento verbal. Así mismo, existe evidencia de que las personas con autismo prefieren estrategias visuales al resolver la RPM y otras tareas cognitivas, a diferencia de las personas con desarrollo típico (Soulières et al., 2009).

Por lo anterior, en este trabajo se explora la posibilidad de resolver la RPM mediante representaciones puramente visuales, sin convertir las imágenes a una representación proposicional. Para esto se desarrollaron dos métodos: uno afín y otro fractal.

Ambos métodos están basados en transformaciones de imágenes que preservan la similitud: identidad (ninguna transformación), reflexión (horizontal o vertical), rotación ortonormal (de 90°, 180° o 270°) y traslaciones. El método fractal utiliza, además, escalamiento contractivo. Existe evidencia de que las personas pueden aplicar algunas operaciones similares a éstas sobre imágenes mentales.

Los dos métodos utilizan un concepto de similitud visual basado en el modelo del cociente (Tversky, 1977):

$$similitud(A,B) = \frac{f(A \cap B)}{f(A \cap B) + \alpha f(A - B) + \beta f(B - A)} \tag{1.1}$$

Donde A y B representan conjuntos de características, f representa alguna función sobre estas características (por ejemplo, un conteo de características), y α y β representan pesos para las partes de los conjuntos A y B que no están en su intersección. Estos pesos se ajustan de tres formas en este trabajo:

- Con $\alpha = 1$ y $\beta = 1$, la Ecuación 1.1 resulta $similitud(A,B) = \frac{f(A \cap B)}{f(A \cup B)}$, que resulta máxima cuando $A = B$.
- Con $\alpha = 1$ y $\beta = 0$, la Ecuación 1.1 resulta $similitud(A,B) = \frac{f(A \cap B)}{f(A)}$, que resulta máxima cuando A es un subconjunto de B.
- Con $\alpha = 0$ y $\beta = 1$, la Ecuación 1.1 resulta $similitud(A,B) = \frac{f(A \cap B)}{f(B)}$, que resulta máxima cuando B es un subconjunto de A.

La primera variante es utilizada en los dos métodos propuestos para identificar imágenes similares. Las siguientes dos son utilizadas en el método afín para capturar la noción de composición de imágenes (suma y resta).

En el método afín cada característica se define como un pixel, y las operaciones de intersección, unión y diferencia entre características son definidas como el producto, máximo y diferencia de valores RGB, respectivamente. El método fractal utiliza características derivadas de una codificación fractal de las imágenes.

Método afín

Este método es similar al propuesto por Lovett et al. (2007) en el sentido de que la matriz es dividida en diferentes analogías. Sin embargo, en este caso se utilizan más analogías y éstas se analizan utilizando sólo operaciones visuales. El método realiza los siguientes pasos:

1. Se divide a la matriz en diferentes analogías de la forma "W es a X, como Y es a Z", donde W, X y Y son celdas definidas y Z es la celda faltante.
2. Para cada analogía, se trata de encontrar una transformación T que permita convertir a W en X, es decir, que aproxime T(W) = X.
3. Se selecciona a la analogía que haya permitido obtener la mayor similitud entre T(W) y X (de acuerdo a la primera variante de la Ecuación 1.1).
4. Se define Z = T(Y) para generar la celda faltante de la matriz.
5. Se selecciona la respuesta más similar a Z (de acuerdo a la primera variante de la Ecuación 1.1).

Al etiquetar las celdas de la matriz como en la Figura 1.17, las transformaciones y analogías consideradas son las descritas en la Tabla 1.1.

A	B
C	X

a)

A	B	C
D	E	F
G	H	X

b)

Figura 1.17. Etiquetas para las celdas de la matriz. a) Matriz de 2x2. b) Matriz de 3x3.

Tabla 1.1. Transformaciones y analogías consideradas por el método afín.

	Transformaciones	Relaciones	
		Matriz de 2x2	Matriz de 3x3
De dos elementos	Identidad Reflexión vertical Reflexión horizontal Rotación de 90° Rotación de 180° Rotación de 270°	AB→CX AC→BX	AC→GX AG→CX BC→HX BH→CX DF→GX DG→FX EF→HX EH→FX GH→HX CF→FX
De tres elementos	Unión Intersección XOR	–	ABC→GHX DEF→GHX ADG→CFX BEH→CFX

Para realizar el paso 2 del procedimiento anterior, se ajusta cada una de las transformaciones listadas para tratar de convertir a W en X. Para ajustar transformaciones de dos elementos se realiza el siguiente procedimiento:

1. Se aplica la transformación a ajustar (T) a W.
2. Se define la traslación (t_x, t_y) que produzca la mayor similaridad entre W y X (de acuerdo a la primera variante de la Ecuación 1.1).
3. Se aplica la traslación (t_x, t_y) a W.
4. Se selecciona la variante de la Ecuación 1.1 que produzca la mayor similitud entre W y X.
    a. Si se eligió la primera variante, la transformación ajustada es:
    $T_a(W) = \left((t_x, t_y) \circ T\right)(W)$.
    b. Si se eligió la segunda variante, la transformación ajustada es:
    $T_a(W) = \left((t_x, t_y) \circ T\right)(W) + (X - W)$,
    donde $+$ y $-$ representan la suma y resta de imágenes.
    c. Si se eligió la tercera variante, la transformación ajustada es:
    $T_a(W) = \left((t_x, t_y) \circ T\right)(W) - (W - X)$.

Para ajustar las transformaciones de tres elementos se sigue el mismo procedimiento, pero la transformación se aplica a los primeros dos elementos de la analogía y se compara con el tercer elemento. Por ejemplo, para ajustar la transformación *unión* con la analogía ABC→GHX, se calcula la unión de las imágenes A y B, y el resultado se compara con la imagen C. Tras ajustar la transformación, ésta puede aplicarse a las imágenes G y H para generar X.

Método fractal

Este método está basado en una codificación fractal de parejas de imágenes. Para problemas con matriz de 2x2 consta de los siguientes pasos:

1. Se divide a la matriz en 8 relaciones. Usando el etiquetado de la Figura 1.17, estas relaciones son: AB–CX, AC–BX, BC–AX, CB–AX y las versiones *inversas* de éstas (BA–XC, CA–XB, CB–XA y BC–XA).
2. Para cada una de las relaciones, y para cada posible respuesta del problema:
   2.1. Se reemplaza X con la posible respuesta.
   2.2. Se calculan dos conjuntos de transformaciones, cada uno para convertir a la primera imagen en la segunda imagen de cada pareja; para esto se utiliza un algoritmo de codificación fractal. Por ejemplo, para la relación AB–CX, se calcula un conjunto de transformaciones que convierte a A en B, y otro conjunto que convierte a C en X.
   2.3. Se calcula la similitud entre los dos conjuntos de transformaciones calculados.
3. El paso anterior asigna 8 valores de similitud a cada posible respuesta (un valor por cada relación). Estos 8 valores se consideran como un vector 8-dimensional $\boldsymbol{v}$, y se calcula su longitud euclidiana $\|\boldsymbol{v}\| = \sqrt{\sum_i v_i^2}$.
4. Se selecciona la respuesta que produjo el vector de mayor longitud.

Para problemas con matriz de 3x3 se sigue el mismo procedimiento, pero la matriz se divide en 48 relaciones en el paso 1. Para calcular la similitud entre conjuntos de transformaciones (paso 2.3) se utiliza la segunda variante de la Ecuación 1.1; esta variante se eligió experimentalmente.

En McGreggor et al. (2010) pueden consultarse los detalles del algoritmo de codificación fractal, así como el procedimiento para calcular la similitud entre los conjuntos de transformaciones generados por este algoritmo.

Resultados y conclusiones

Ambos métodos se evaluaron con los 60 problemas de la RPM estándar. Cada problema fue escaneado y dividido de modo que cada celda de la matriz y cada posible respuesta estuviera en un archivo separado. Las imágenes no recibieron ningún otro procesamiento, de modo que contenían ruido y desalineaciones. El método afín resolvió 35 problemas y el fractal 32, como muestra la Figura 1.18.

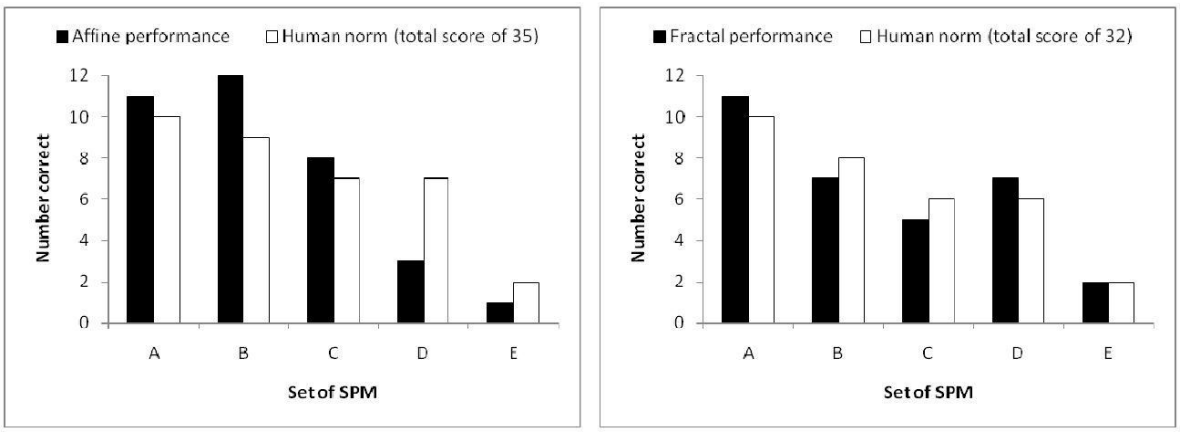


Figura 1.18. Desempeño de los métodos afín (izquierda) y fractal (derecha) (Kunda et al., 2010).

En esta figura, las barras negras muestran el desempeño de los métodos en cada sección de la prueba; las barras blancas muestran el desempeño típico de las personas que obtuvieron puntajes totales de 35 (izquierda) o 32 (derecha). Se observa que el método fractal sigue el desempeño de las

personas con una diferencia máxima de un punto, mientras que el método afín presenta una diferencia máxima de cuatro puntos (en la sección D). De acuerdo al manual de la RPM estándar (Raven et al., 2004), si esta diferencia es mayor a dos puntos entonces el puntaje total no necesariamente refleja las habilidades cognitivas del participante (debido a que posiblemente adivinó respuestas, respondió aleatoriamente, o comprendido incorrectamente las instrucciones de la prueba).

Los autores presentan las siguientes conclusiones:

- Los dos métodos presentados lograron resolver más de la mitad de los problemas de la RPM estándar, partiendo de imágenes sin procesar y utilizando sólo estrategias visuales.
- Lo anterior apoya el uso de transformaciones de similitud para manipular imágenes, y el uso del modelo del cociente (Ecuación 1.1) para comparar imágenes.
- Los resultados obtenidos muestran que es factible realizar razonamiento en el nivel perceptual en sistemas de inteligencia artificial.
- Los métodos presentados pueden brindar nuevas ideas para el análisis del procesamiento visual.

Rasmussen et al. (2011)

Este trabajo presenta un modelo basado en *neuronas de impulsos* (spiking neurons) para resolver la RPM. De acuerdo a los autores, el modelo no se limita a aplicar reglas definidas manualmente, sino que genera reglas a partir de las figuras de la matriz, por lo que es el primero en abordar el proceso de inducción de reglas en la RPM. Dado que la prueba pretende medir la *habilidad eductiva* y no la *habilidad reproductiva*[7], el elemento crucial en su resolución es la inducción de reglas y no la aplicación de reglas predefinidas. El modelo reproduce, además, varios efectos cognitivos observados en las personas que resuelven la prueba, como el aprendizaje y la variabilidad en el desempeño.

Representación vectorial

Los problemas de la RPM se representaron simbólicamente usando la arquitectura VSA (Vector Symbolic Architecture) definida por Gayler (2003), en la cual la información se representa mediante vectores y se definen operaciones matemáticas para combinar estos vectores. Para implementar esta representación se definieron cuatro elementos:

- Una *operación vinculante* $\otimes$, que permite enlazar dos vectores y que se definió como una convolución circular: $\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b}$, donde $\boldsymbol{a}$, $\boldsymbol{b}$ y $\boldsymbol{c}$ son vectores n-arios y $c_j = \sum_{k=0}^{n-1} \left(a_k b_{(j-k) \bmod n} \right)$.
- Una *operación de superposición* $+$, que permite definir conjuntos de vectores y que se definió como una suma entre vectores.
- Un concepto de *vector de transformación* $\boldsymbol{t}$ entre dos vectores $\boldsymbol{a}$ y $\boldsymbol{b}$, definido como: $\boldsymbol{a} \otimes \boldsymbol{t} = \boldsymbol{b}$, o bien $\boldsymbol{t} = \boldsymbol{a}' \otimes \boldsymbol{b}$, donde $\boldsymbol{a}'$ es el vector inverso aproximado de $\boldsymbol{a}$; es decir, el vector $\boldsymbol{t}$ permite convertir al vector $\boldsymbol{a}$ en el vector $\boldsymbol{b}$ mediante la convolución circular
- Un *vocabulario*, que es un conjunto de vectores que se utilizan como bloques constructores y que se definieron aleatoriamente. Por ejemplo, el vector [0.1, –0.35, 0.17, …] podría usarse para representar *círculo*. La dimensionalidad de estos vectores está determinada por el

[7] Véase *La teoría de la inteligencia de Spearman* en la Sección 1.1.

número de vectores en el vocabulario: para diferenciar más vectores se requieren más dimensiones.

Con estos elementos, cada celda de la matriz se representa como un conjunto (superposición) de pares atributo-valor, donde cada par se representa mediante dos vectores del vocabulario enlazados (vinculados). Por ejemplo, la Figura 1.19 se representaría como el vector resultante de calcular $(forma \otimes círculo + cantidad \otimes tres + color \otimes negro + orientación \otimes horizontal + relleno \otimes sólido + \cdots)$, donde cada atributo y valor es un vector del vocabulario.

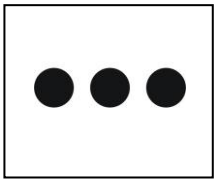


Figura 1.19. Tres círculos negros alineados horizontalmente.

Algunas ventajas de la arquitectura VSA son:

- Permite representar información de cualquier complejidad.
- A comparación de las imágenes sin procesar, requiere menos recursos, es más fácil de procesar matemáticamente y permite aislar al sistema de los detalles de la representación visual.

Representación neuronal

Para representar y procesar los vectores en una red de neuronas de impulsos, se utilizaron las técnicas del entorno NEF (Neural Engineering Framework) definido por Eliasmith et al. (2003). La Figura 1.20 ilustra los procesos de codificación, transformación y decodificación involucrados.

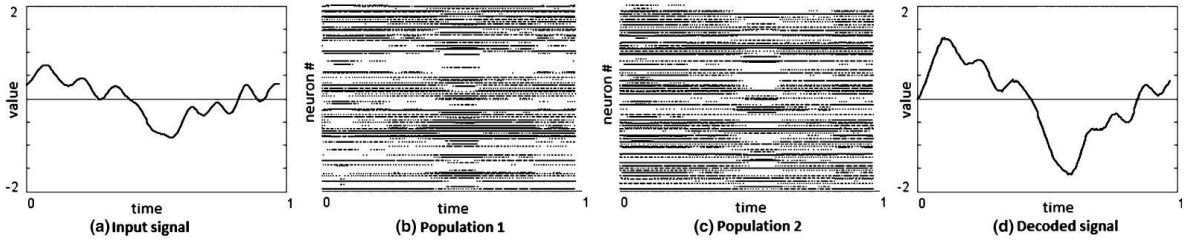


Figura 1.20. Elementos del entrono NEF (Rasmussen et al., 2011). (a) a (b): Codificación desde una señal de entrada hacia la actividad de una población de neuronas (Ecuación 1.2). (b) a (c): Transformación (en este caso, multiplicación por dos) del valor representado (Ecuación 1.5). (c) a (d): Decodificación desde la actividad de una población de neuronas hacia una señal de salida (Ecuación 1.4). En las imágenes (b) y (c), cada línea horizontal representa la actividad de una neurona, y cada punto negro representa un impulso.

Para codificar un vector $\boldsymbol{x}$ en el tren de impulsos de una población de neuronas $\boldsymbol{a}$ se utiliza la siguiente ecuación:

$$a_i(\boldsymbol{x}) = G_i\left[\alpha_i \widetilde{\phi}_i \boldsymbol{x} + J_i^{bias}\right] \tag{1.2}$$

Donde $a_i(\boldsymbol{x})$ es el tren de impulsos de la neurona i. G_i Es un modelo de neurona que toma como entrada una corriente eléctrica (el valor entre paréntesis cuadrados) y devuelve como salida un tren

de impulsos en el tiempo. α_i Es una ganancia que se utiliza para generar variedad en los impulsos de las neuronas en una población. $\tilde{\phi}_i$ Representa el estímulo preferido por la neurona i, es decir, el estímulo para el cual la neurona genera más impulsos. J_i^{bias} Es una constante que representa corrientes residuales generadas por los procesos intrínsecos de la neurona o por la actividad de las demás neuronas en la red.

Para el parámetro G_i se utilizó el modelo de neurona de *integración y disparo con fuga* (leaky integrate and fire, o LIF)[8]:

$$I(t) - \frac{V_m(t)}{R_m} = C_m \frac{dV_m(t)}{dt} \tag{1.3}$$

Donde $I(t)$ es la corriente que entra a la neurona, $V_m(t)$ es el voltaje a través de su membrana, R_m es la resistencia de su membrana y C_m es la capacitancia de su membrana.

Mientras que la Ecuación 1.2 se utiliza para codificar un vector en un conjunto de trenes de impulsos, la siguiente ecuación se utiliza para obtener un vector a partir de un conjunto de trenes de impulsos:

$$\boldsymbol{y} = \sum_i \big(h * a_i(\boldsymbol{x_i})\big)\phi_i \tag{1.4}$$

Donde $\boldsymbol{y}$ es el vector obtenido. $*$ Representa convolución estándar (no circular). h Es un modelo que, al convolucionarse con un tren de impulsos, permite estimar la corriente total generada por estos impulsos en la neurona que los recibe. Cada $a_i(\boldsymbol{x_i})$ es el tren de impulsos de una neurona, calculado mediante la Ecuación 1.2. ϕ_i Es un decodificador lineal óptimo, que se calcula analíticamente para obtener la mejor representación linal del vector de entrada original $\boldsymbol{x_i}$.

Finalmente, para realizar las operaciones definidas en VSA (vinculación y superposición), se definen transformaciones lineales de la forma $\boldsymbol{z} = C_1\boldsymbol{x} + C_2\boldsymbol{y}$, donde $\boldsymbol{x}$, $\boldsymbol{y}$ y $\boldsymbol{z}$ son vectores y C_1 y C_2 son matrices. Para cada una de estas transformaciones, cada vector ($\boldsymbol{x}$, $\boldsymbol{y}$ y $\boldsymbol{z}$) se representa con una población de neuronas ($\boldsymbol{a}$, $\boldsymbol{b}$ y $\boldsymbol{c}$, respectivamente), y las matrices se transforman analíticamente en pesos sinápticos entre las neuronas (no se aplica un algoritmo de entrenamiento a la red neuronal). Con base en lo anterior, la actividad de la población de salida $\boldsymbol{c}$ se calcula con la siguiente ecuación:

$$c_k(\boldsymbol{x}, \boldsymbol{y}) = G_k\big[\textstyle\sum_i \omega_{ki} a_i(\boldsymbol{x}) + \sum_j \omega_{kj} b_j(\boldsymbol{y}) + J_k^{bias}\big] \tag{1.5}$$

Donde cada a_i, b_j y c_k es el tren de impulsos de una neurona de la población $\boldsymbol{a}$, $\boldsymbol{b}$ y $\boldsymbol{c}$, respectivamente. G_k Es un modelo de neurona que recibe una corriente eléctrica y devuelve un tren de impulsos (como en la Ecuación 1.2). ω_{ki} Y ω_{kj} son pesos sinápticos calculados analíticamente a partir de las matrices C_1 y C_2:

$$\omega_{ki} = \alpha_k \langle \tilde{\phi}_k C_1 \phi_i^x \rangle_m \tag{1.6}$$
$$\omega_{kj} = \alpha_k \langle \tilde{\phi}_k C_2 \phi_j^y \rangle_m \tag{1.7}$$

Donde α_k es la ganancia de la neurona c_k. $\tilde{\phi}_k$ Es el estímulo preferido por la neurona c_k. ϕ_i^x Es el decodificador lineal óptimo de la neurona i. Y ϕ_j^y es el decodificador lineal óptimo de la neurona j.

[8] Para una descripción del modelo LIF véase Dayan et al. (2001), sección 5.4.

Para calcular diferentes transformaciones sólo se necesita cambiar las matrices C en las ecuaciones 1.6 y 1.7, lo cual permite realizar todos los cálculos lineales necesarios en este modelo. Para una descripción más detallada de el entorno NEF, véase Eliasmith et al. (2003) y Eliasmith (2005).

Inducción de reglas

Una vez que cada celda de una matriz se ha representado como un vector (usando la arquitectura VSA), las reglas de la matriz pueden representarse como transformaciones entre estos vectores, usando el concepto de *vector de transformación* definido anteriormente. Por ejemplo, si el vector $\boldsymbol{x}$ representa una celda que contiene un cuadrado, y el vector $\boldsymbol{y}$ representa una celda que contiene dos cuadrados, entonces el vector $\boldsymbol{t} = \boldsymbol{x}' \otimes \boldsymbol{y}$ (o bien $\boldsymbol{x} \otimes \boldsymbol{t} = \boldsymbol{y}$) permite convertir al vector $\boldsymbol{x}$ en el vector $\boldsymbol{y}$, y representa la regla "un cuadrado se convierte en dos cuadrados". Sin embargo, esta regla es específica para las dos celdas representadas por $\boldsymbol{x}$ y $\boldsymbol{y}$. Para inferir reglas generales y resolver cada problema de la RPM se realiza el siguiente procedimiento:

1. Se define un vector de transformación $\boldsymbol{t_i}$ para cada par de celdas adyacentes $\boldsymbol{x_i}$ y $\boldsymbol{y_i}$ en la misma fila de la matriz, calculando $\boldsymbol{t_i} = \boldsymbol{x_i'} \otimes \boldsymbol{y_i}$ (donde $\boldsymbol{x_i'}$ es el vector inverso aproximado de la celda $\boldsymbol{x_i}$).
2. Se calcula el promedio de todos los vectores de transformación definidos: $\boldsymbol{t} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{t_i}$.
3. Se genera la celda faltante de la matriz, calculando $\boldsymbol{y} = \boldsymbol{x} \otimes \boldsymbol{t}$, donde $\boldsymbol{x}$ es la celda adyacente a la celda faltante (en su misma fila).
4. Se compara la celda generada con cada una de las posibles respuestas, calculando $\boldsymbol{y} \cdot \boldsymbol{r_i}$, donde $\boldsymbol{r_i}$ es el vector de cada posible respuesta y · es el producto punto entre vectores.
5. Se selecciona la respuesta cuyo vector produjo el máximo $\boldsymbol{y} \cdot \boldsymbol{r_i}$.

Todas las operaciones entre vectores se realizan mediante poblaciones de neuronas, como se muestra en la Figura 1.21.

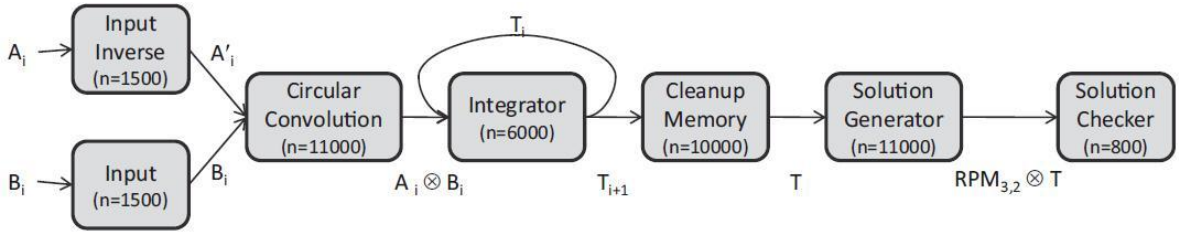


Figura 1.21. Esquema del modelo de inducción de reglas (Rasmussen et al., 2011). n Representa el número aproximado de neuronas en cada módulo.

En la Figura 1.21, A_i y B_i son vectores que representan celdas adyacentes en la misma fila de la matriz. A_i' Es el vector inverso aproximado de A_i. T_i Representa cada vector de transformación calculado, y T es el promedio de estos vectores. $RPM_{3,2}$ Es la celda a la izquierda de la celda faltante (en matrices de 3x3). $RPM_{3,2} \otimes T$ Es la solución generada por el sistema, que luego se compara con las posibles respuestas para elegir la respuesta más similar. El módulo *Cleanup Memory* se describe a continuación.

Aprendizaje

Una *memoria de limpieza* (cleanup memory) es un sistema que almacena ciertos valores y, al recibir una versión distorsionada de alguno de ellos, devuelve una versión limpia del mismo. En este

modelo se incluye una memoria de limpieza que almacena las reglas generadas previamente por el sistema. Esta memoria se mejora gradualmente usando dos mecanismos:

- Si la memoria recibe una regla que no reconoce, la regla es almacenada.
- Si la memoria recibe una regla que sí reconoce, la regla es utilizada para refinar la regla almacenada correspondiente.

La memoria de limpieza permite mejorar el desempeño del sistema y modelar los efectos del aprendizaje en la RPM (Figura 1.22). Además crea un puente entre este modelo, que genera reglas dinámicamente, y los modelos que utilizan un conjunto de reglas predefinidas. La memoria de limpieza no proporciona reglas directamente, sino que requiere que el sistema induzca estas reglas para luego compararlas con el contenido de la memoria. Por otro lado, el sistema aún puede beneficiarse de las reglas que ha aprendido en el pasado.

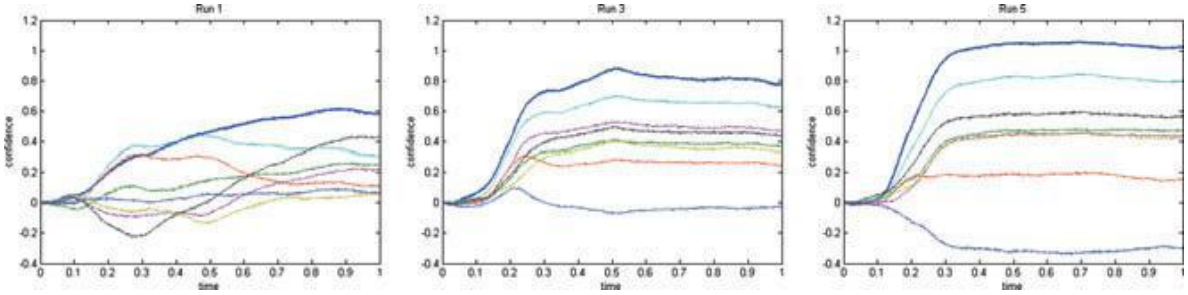


Figura 1.22. El aprendizaje en el sistema (Rasmussen et al., 2011). Se introdujeron sucesivamente 5 problemas (con matriz de 3x3) al sistema, los cuales tenían diferentes figuras pero requerían las mismas reglas para resolverse. Cada imagen describe la resolución del 1°, 3° y 5° problema (de izquierda a derecha). Las gráficas muestran la confianza del sistema en cada una de las 8 posibles respuestas conforme se realizaba el proceso de inducción de reglas. La confianza se muestra en el eje vertical y el tiempo en el eje horizontal. Se observa que el sistema discrimina mejor y en menos tiempo las respuestas del último problema presentado.

Procesos de alto nivel

Para coordinar los procesos descritos anteriormente, el modelo incluye un *sistema de estrategia*. Este sistema se programó sin usar redes neuronales y tiene las siguientes funciones:

- Proporciona las entradas a los módulos neuronales.
- Evalúa el éxito de las reglas generadas, para definir si un problema ya puede resolverse o si se debe continuar con el procesamiento.
- Selecciona la respuesta final cuando los módulos neuronales generan varias reglas (en problemas que requieren la inducción de varias reglas).

El sistema de estrategia tiene dos parámetros:

- La dimensionalidad de los vectores utilizados para codificar los problemas. Este parámetro de bajo nivel modela las diferencias en la capacidad de procesamiento de las personas.
- Un valor umbral, que se utiliza para decidir si ya se han generado las reglas necesarias para resolver un problema o si se debe continuar con el procesamiento. Este parámetro de alto nivel modela las diferencias en las estrategias que siguen las personas al resolver la RPM.

La Figura 1.23 muestra cómo estos parámetros afectan el desempeño del sistema.

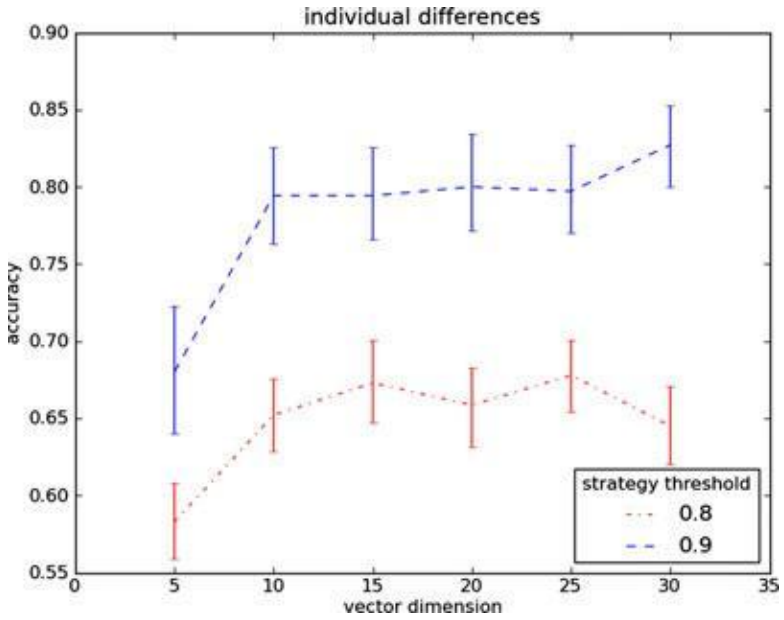


Figura 1.23. Efecto de los parámetros de alto y bajo nivel en el desempeño del sistema (Rasmussen et al., 2011). El eje vertical muestra el desempeño y el horizontal la dimensionalidad de los vectores utilizados. La gráfica superior corresponde a un umbral de 0.9 y la inferior a un umbral de 0.8. Se observa que el desempeño del sistema es variable (no determinista), pues la desviación estándar del desempeño es 0.13 en promedio.

Resultados y conclusiones

Los autores presentan las siguientes conclusiones:

- Se presentó un modelo novedoso basado en redes neuronales de impulsos para la generación inductiva de reglas, y se aplicó este modelo a la RPM.
- El éxito del sistema es demostrado por su capacidad para encontrar reglas generales que permiten resolver estas matrices, y por su capacidad para reproducir los efectos observados en las personas al resolver la prueba, como el aprendizaje, el desempeño no determinista y las diferencias cuantitativas (capacidad de procesamiento) y cualitativas (estrategia).
- Estos resultados demuestran el potencial de los modelos neuronales para lograr una mejor comprensión de los procesos de inducción humanos.

Los autores no indican cuántos problemas fueron resueltos por el sistema.

Análisis

A continuación se comparan los programas presentados, tomando como base siete aspectos:

1. Cómo representaron los problemas de la RPM.
2. Cómo realizaron razonamiento por analogía.
3. Cómo modelaron la inducción de reglas.
4. Cómo modelaron el aprendizaje.
5. Cómo seleccionaron la respuesta en cada problema.
6. Qué resultados obtuvieron.
7. Qué conclusiones e implicaciones presentaron.

El programa desarrollado por Bringsjord et al. (2003) casi no se menciona, pues se desconocen los detalles de su implementación y sus resultados.

1. Representaciones

Siete de los nueve programas presentados partieron de representaciones simbólicas y hechas a mano de los problemas de la RPM: Carpenter et al. (dos programas), Bringsjord et al., Lovett et al. (dos programas), Cirillo et al., y Rasmussen et al. En estos programas, cada problema se describe utilizando un conjunto restringido de figuras y atributos básicos, como *círculo*, *relleno sólido* y *color negro*. De estos siete programas:

- Dos siguieron un formato estándar para describir imágenes vectoriales: WMF en Lovett et al. (2 programas). Los demás definieron sus propios conjuntos de figuras y atributos.
- Tres utilizaron representaciones jerárquicas, que organizan a los elementos de cada celda de la matriz en varios niveles (como líneas, figuras y grupos): Lovett et al. (2 programas), y Cirillo et al.

Sólo los dos programas de Kunda et al. utilizaron representaciones subsimbólicas de los problemas, pues partieron de imágenes sin procesar (representadas con pixeles). Además, estos programas fueron los únicos que resolvieron la sección A de la RPM estándar, la cual sólo contiene problemas basados en una sola imagen (Figura 1.1).

Aunque varios de los trabajos presentados consideran que el procesamiento de imágenes no desempeña un papel importante al resolver la prueba, en Meo et al. (2007) se propone que la familiaridad de las figuras de la RPM está vinculada a la carga que provocan estas figuras en la memoria de trabajo de las personas; además, esta familiaridad está asociada a la dificultad para encontrar las correspondencias entre los elementos de la matriz, y por lo tanto influye en la dificultad de cada problema.

2. Razonamiento por analogía

Aunque los programas utilizaron representaciones y estrategias diferentes, todos incluyeron procesos que les permitieron realizar razonamiento por analogía:

- Procesos para encontrar las correspondencias entre los elementos de la matriz.
    - En Carpenter et al. se utilizaron tres mecanismos: heurística de nombres, heurística de sobrantes y reglas identificadas. Además, se definieron cinco tipos de reglas y las matrices se procesaron por filas.
    - En Lovett et al. (2007) se utilizó el sistema sKEA (Sketching Knowledge Entry Associate) y el sistema SME (Structure Mapping Engine, que es un modelo de analogía en sí mismo), y se dividió a cada matriz en varios problemas de analogía.
    - En Lovett et al. (2010) se utilizó el sistema CogSketch y el sistema SME, se definieron cuatro *estrategias* y las matrices se procesaron por filas.
    - En Cirillo et al. se utilizó una representación jerárquica y se definieron siete *patrones*. Además, las matrices se procesaron por filas y por columnas.
    - En Kunda et al. (método afín) se dividió a cada matriz en varios problemas de analogía y se definió un conjunto de transformaciones visuales.
    - En Kunda et al. (método fractal) se dividió a cada matriz en varias relaciones.

    - En Rasmussen et al. las matrices se procesaron por pares de celdas adyacentes en la misma fila.
- Procesos para comparar los elementos correspondientes e identificar patrones de variación.
    - En Carpenter et al. se implementó un módulo de *comparación por pares* y se definieron cinco tipos de reglas.
    - En Lovett et al. (2007) se utilizó el sistema sKEA y el sistema SME.
    - En Lovett et al. (2010) se utilizó el sistema CogSketch y el sistema SME, y se definieron cuatro *estrategias*.
    - En Cirillo et al. se definieron siete *patrones*, y se utilizó una representación jerárquica que permite comparar elementos en diferentes niveles de organización.
    - En Kunda et al. (método afín) se definió un concepto de similitud visual (Ecuación 1.1) y un conjunto de transformaciones visuales.
    - En Kunda et al. (método fractal) se definió un concepto de similitud visual (Ecuación 1.1) y un conjunto de transformaciones fractales.
    - En Rasmussen et al. se definió un concepto de transformación entre vectores y se utilizó el producto punto para comparar vectores.
- Procesos para generalizar patrones de variación.
    - En Carpenter et al. se compararon y generalizaron las reglas identificadas. Además, los cinco tipos de reglas utilizados representan patrones de variación generales que se definieron manualmente.
    - En Lovett et al. (2007) se compararon conjuntos de diferencias y similitudes mediante SME. En este caso, las similitudes entre estos conjuntos definen implícitamente patrones de variación generales.
    - En Lovett et al. (2010) se compararon y generalizaron *patrones de variación*. Además, las cuatro *estrategias* utilizadas representan patrones de variación generales que se definieron manualmente.
    - En Cirillo et al. se utilizaron siete patrones de variación generales definidos manualmente.
    - En Kunda et al. (método afín) se utilizaron transformaciones visuales que representan patrones de variación generales definidos manualmente.
    - En Kunda et al. (método fractal) se compararon conjuntos de transformaciones fractales. Las similitudes entre estos conjuntos definen implícitamente patrones de variación generales.
    - En Rasmussen et al. se promediaron transformaciones específicas, calculadas entre pares de celdas, para tratar de obtener transformaciones generales.

Lo anterior, sumado a lo presentado en la Sección 1.1 y a varios estudios de la analogía (como Doumas et al., 2008; y Gentner et al., 2001), muestra una estrecha relación entre la prueba de Raven, el razonamiento por analogía y la inteligencia:

- La prueba de Raven permite analizar aspectos importantes de la inteligencia.
- El razonamiento por analogía permite resolver la prueba de Raven.
- El razonamiento por analogía permite el desarrollo y ejecución de una amplia gama de procesos inteligentes.

3. Inducción de reglas

La RPM pretende medir la *habilidad eductiva* (la habilidad para descubrir relaciones a partir de información nueva) y no la *habilidad reproductiva* (la habilidad para aplicar información que se ha

aprendido)[9]; además, la novedad de los problemas es determinante para la validez de la prueba (Carpenter et al., 1990). Por tanto, el proceso crucial en la resolución de la RPM es la inducción de reglas y no la aplicación de reglas aprendidas (Rasmussen et al., 2011).

Inducir reglas significa crear reglas generales a partir de ejemplos específicos. Sin embargo, en varios de los programas presentados se definieron reglas generales de forma manual:

- En Carpenter et al. se definieron cinco tipos de reglas.
- En Lovett et al. (2010) se definieron cuatro estrategias.
- En Cirillo et al. se definieron siete patrones.
- En Kunda et al. (método afín) se definieron nueve transformaciones visuales.

En lugar de un proceso de inducción, estos programas más bien realizan un proceso de deducción: adaptar reglas generales a casos específicos. Los programas que sí modelaron un proceso de inducción fueron:

- Lovett et al. (2007), que genera reglas a partir de mapeos de SME.
- Kunda et al. (método fractal), que genera reglas a partir de transformaciones fractales.
- Rasmussen et al., que genera reglas a partir de transformaciones vectoriales.

Sin embargo, como muestra la Figura 1.24, los programas que lograron resolver los problemas más difíciles[10] fueron los que utilizaron reglas predefinidas:

- El programa Betteraven de Carpenter et al. resolvió 32 de los 48 problemas de la RPM avanzada.
- El programa de Lovett et al. (2010) resolvió 44 de 48 problemas en las secciones B a E de la RPM estándar. Los autores no reportaron el puntaje en cada sección, pero la Figura 1.24 muestra una distribución estimada con base en la dificultad de los problemas.
- El programa de Cirillo et al. resolvió 28 de 36 problemas en las secciones C, D y E de la RPM estándar.
- El programa del método afín de Kunda et al. resolvió 35 de 60 problemas en la RPM estándar. Aunque este programa resolvió pocos problemas en las últimas secciones de la prueba, su puntaje total fue mejor que el del programa del método fractal de los mismos autores. Además, a diferencia de los otros programas con reglas predefinidas, el del método afín partió de imágenes sin procesar y no de representaciones simbólicas hechas a mano.

En cambio, los programas que sí modelaron la inducción de reglas lograron resolver menos problemas difíciles:

- El programa de Lovett et al. (2007) sólo resolvió 22 de 24 problemas en las secciones B y C de la RPM estándar.
- El programa del método fractal de Kunda et al. resolvió 32 de 60 problemas en la RPM estándar, 3 problemas menos que el programa del método afín. Aunque no resolvió más de 7 problemas en las secciones B a E, su desempeño en las secciones D y E fue mejor que el del método afín.
- Se desconoce el desempeño del programa de Rasmussen et al. Sin embargo, con base en el concepto de transformación que se definió en este trabajo, el programa sólo puede procesar relaciones entre pares de celdas de la matriz. Varios problemas de la RPM requieren

[9] Véase *La teoría de la inteligencia de Spearman* en la Sección 1.1.
[10] Como se mencionó en la Sección 1.1, la dificultad de los problemas tiende a aumentar conforme se avanza en la prueba.

considerar tres celdas simultáneamente (Carpenter et al., 1990), por lo que el desempeño de este programa fue probablemente bajo en las últimas dos secciones de la prueba.

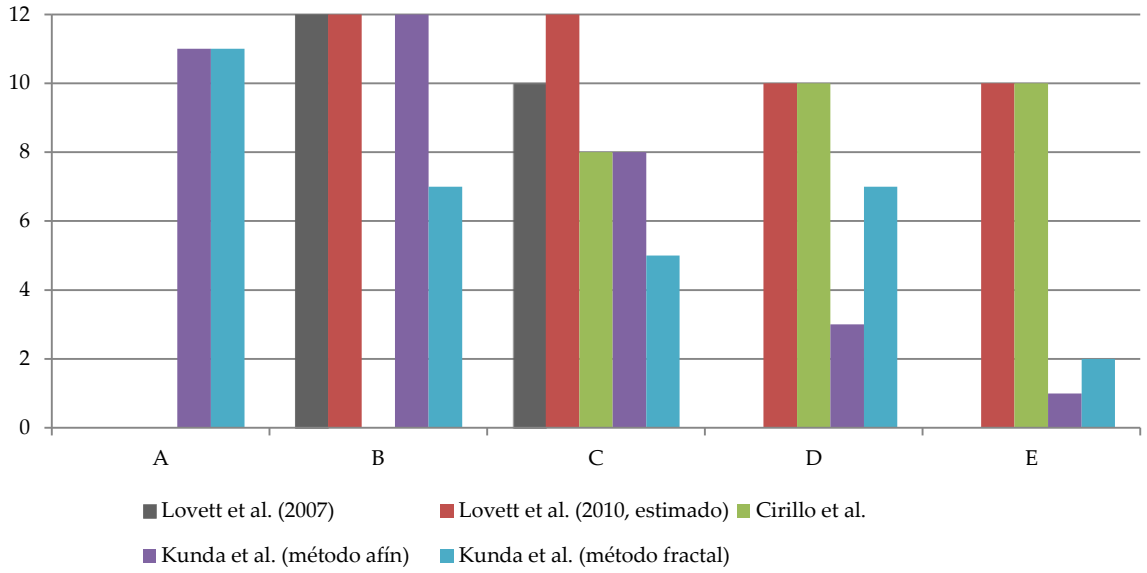


Figura 1.24. Desempeño de los cinco programas que resolvieron la RPM estándar[11].

La dificultad de modelar la inducción de reglas proviene de un problema fundamental en la ciencia cognitiva. Aunque ha sido posible modelar el razonamiento no estructurado (subsimbólico, perceptual y concreto) característico de los niños pequeños, y también el razonamiento estructurado (simbólico, interpretativo y abstracto) de los adolescentes y adultos, no se cuenta con modelos satisfactorios de cómo el primero se convierte en el segundo, es decir, de cómo las personas desarrollan representaciones estructuradas a partir de ejemplos no estructurados (Doumas et al., 2008).

En línea con lo anterior está el hecho de que sólo dos de los programas presentados partieron de representaciones no estructuradas, y el hecho de que los programas que no modelaron el proceso de inducción obtuvieron mejores puntajes.

4. Aprendizaje

Sólo el programa presentado por Rasmussen et al. modeló los efectos del aprendizaje en la RPM. Esto refleja una tendencia general en la literatura, pues casi toda la investigación sobre la prueba (enfocada a las diferencias entre las personas) asume que el aprendizaje no influye durante la resolución de la RPM (Verguts et al., 2002).

Aunque en el resto de los programas no se modeló el aprendizaje, éste tiene un papel importante cuando las personas resuelven la prueba (Verguts et al., 2002):

- Existe un efecto de *inercia mental* (Einstellung effect), por el cual las personas aprenden reglas conforme avanzan en la prueba, y tratan de aplicar estas mismas reglas al abordar problemas subsecuentes.

[11] Para una gráfica imprimible en blanco y negro, consulte el Apéndice A.

- El efecto anterior se debilita en cuanto algún problema no satisface las reglas aprendidas hasta el momento (set-breaking effect).
- Las personas tratan de aplicar con más frecuencia las reglas aprendidas recientemente (recency effect).
- El hecho de que la dificultad de los problemas tiende a incrementarse gradualmente facilita la resolución de la prueba (sequence effect).

Este último punto está relacionado con la noción de *alineación progresiva* (progressive alignment) en el aprendizaje de conceptos. Ésta se refiere a un proceso de entrenamiento en el cual se inicia por comparar ejemplos muy similares de un concepto, y progresivamente se comparan ejemplos más distantes. Este proceso es sobresaliente porque permite el aprendizaje de conceptos sin retroalimentación, y en un menor tiempo que si los ejemplos se presentan desordenados (Doumas et al., 2008).

Los efectos observados por Verguts et al. (2002) se desprenden de experimentos realizados con y sin retroalimentación al resolver problemas de la RPM. Sin embargo, este estudio observa que las posibles respuestas que incluye cada problema proporcionan una forma de retroalimentación implícita, ya que pueden ayudar a las personas a confirmar si su razonamiento es correcto.

5. Selección de la respuesta

En los programas presentados se utilizaron dos estrategias diferentes para seleccionar la respuesta de cada problema:

a) Generar la celda faltante de la matriz y luego compararla con las posibles respuestas: Carpenter et al. (dos programas), Cirillo et al., Kunda et al. (método afín) y Rasmussen et al.
b) Insertar cada posible respuesta en la matriz para asignarle un puntaje: Lovett et al. (dos programas) y Kunda et al. (método fractal).

De acuerdo a Carpenter et al. (1990), las personas con puntaje alto tienden a utilizar la primera estrategia, pues generan la celda faltante (total o parcialmente) antes de analizar las posibles respuestas. Sin embargo, los programas obtuvieron buenos resultados con ambas estrategias, como lo demuestra el programa de Lovett et al. (2010) (Figura 1.25).

6. Resultados

La Figura 1.25 muestra los resultados de los cinco programas que resolvieron la RPM estándar (también mostrados en la Figura 1.24). Los dos programas de Carpenter et al. no se muestran porque se aplicaron a la RPM avanzada (como se mencionó en la Sección 1.2, Fairaven resolvió 23 de 34 problemas, y Betteraven resolvió 32 de 34).

Por tanto, los programas con mejor desempeño son Betteraven de Carpenter et al. y el programa de Lovett et al. (2010). Sin embargo, la parte más significativa de los programas presentados no es sólo el puntaje final que obtuvieron, sino también las metodologías que utilizaron y las conclusiones que permitieron obtener.

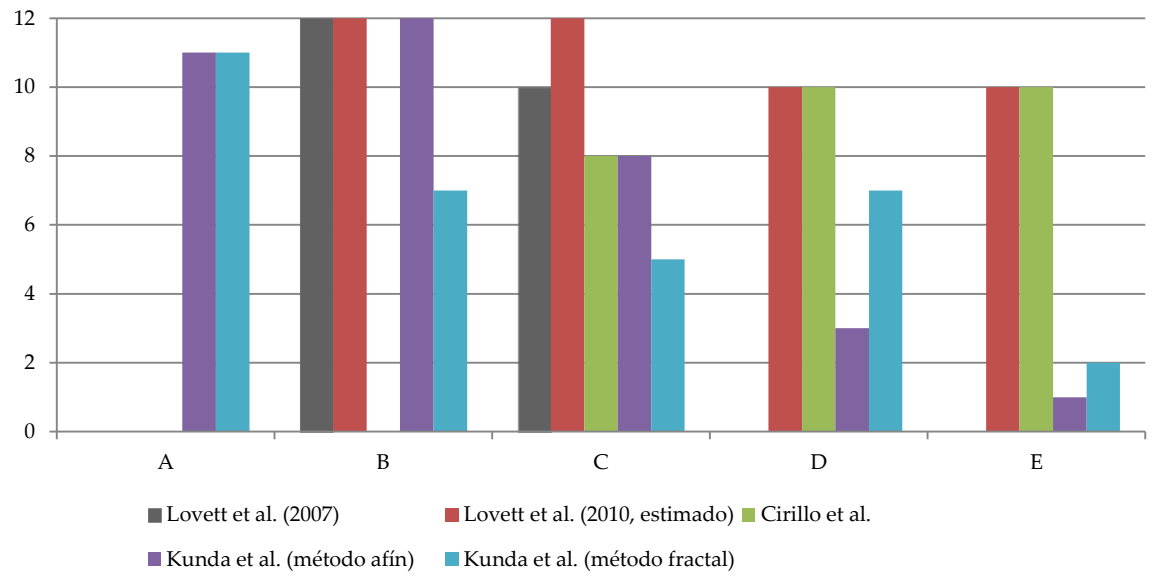


Figura 1.25. Desempeño de los cinco programas que resolvieron la RPM estándar[12].

7. Conclusiones

Los trabajos presentados muestran que al desarrollar programas que resuelvan la RPM es posible investigar cuatro áreas de interés:

- Las características de la RPM, los procesos cognitivos que involucra y los elementos que determinan la dificultad de los problemas.
- Las diferencias entre las personas al resolver la prueba, tanto cualitativas (en estrategia) como cuantitativas (en capacidad de procesamiento).
- Los procesos involucrados en la inteligencia general, como la habilidad para realizar abstracciones, manejar jerarquías de objetivos y realizar analogías.
- La aplicabilidad de diferentes técnicas de inteligencia artificial, simbólicas y subsimbólicas, para representar y manipular información visual, así como para modelar el razonamiento por analogía, la inducción de reglas y el aprendizaje de relaciones.

1.3 Teoría para la síntesis de conceptos geométricos

El programa que se desarrolló en este proyecto está basado en la teoría propuesta por Pineda (2007) para la síntesis de conceptos geométricos. En esta sección se describe brevemente esta teoría, la cual tiene las siguientes características:

- Involucra cuatro componentes principales: principios de conservación, esquemas de acción, la descripción de abstracciones geométricas y la reinterpretación de diagramas.
- Define un proceso constructivo que permite sintetizar una función en el dominio geométrico, la cual puede representar un concepto o teorema geométrico.
- El proceso constructivo consiste en una derivación diagramática en la cual la representación (sintaxis) y la interpretación (semántica) son sintetizadas simultáneamente.

[12] Para una gráfica imprimible en blanco y negro, consulte el Apéndice A

En Pineda (2007) se aplicó esta teoría para sintetizar el teorema de Pitágoras y un teorema sobre la suma de los números impares. Además, la teoría se implementó en un programa prototipo interactivo que asiste al usuario en la síntesis de conceptos geométricos.

Principios de conservación y esquemas de acción

Los principios de conservación son conceptos que permiten saber si un objeto conserva una propiedad tras aplicarle una transformación. Los esquemas de acción, por su parte, representan transformaciones que pueden aplicarse a un objeto en relación con un contexto geométrico. Por ejemplo, la Figura 1.26 muestra un esquema de acción que, al aplicarse en el contexto geométrico de la izquierda, produce el de la derecha y conserva el área total de las figuras.

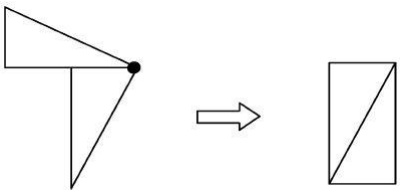


Figura 1.26. Esquema de acción que conserva el área total de las figuras (Pineda, 2007).

Las características de este esquema son:

- Puede aplicarse en cualquier *contexto geométrico*, siempre que éste contenga dos triángulos alineados como se muestra en la imagen izquierda de la Figura 1.26.
- Usa como *foco de atención* a uno de los triángulos (en la figura se utiliza al superior).
- Usa como *pivote* al vértice común de los dos triángulos.
- Consiste en girar al triángulo en el *foco de atención*, usando como eje el *pivote*, de modo que su hipotenusa se alinee con la del otro triángulo.
- Satisface el *principio de conservación de área*.

Los esquemas de acción que conservan cierta propiedad están asociados a principios de conservación. Además, mientras que la aplicación de un esquema de acción produce cambios en un contexto geométrico concreto (sintaxis), la aplicación de un principio de conservación representa una interpretación (semántica) de naturaleza abstracta. Los esquemas de acción pueden consistir en una sola operación, como el esquema de la Figura 1.26, o en varias operaciones, pero siempre son considerados como procesos indivisibles (holísticos).

Un conjunto de principios de conservación y esquemas de acción, y las relaciones entre éstos, representa el conocimiento primitivo a partir del cual pueden sintetizarse conceptos geométricos. La interacción entre estos componentes produce una secuencia de diagramas, y una función en el dominio geométrico que representa el concepto expresado por estos diagramas. Si la secuencia de diagramas producida representa a una clase abstracta de objetos o configuraciones, entonces la función producida representa una relación general o un teorema geométrico.

Lenguaje de representación geométrica

Para expresar esta teoría se definió un lenguaje de representación geométrica, el cual permite especificar configuraciones geométricas concretas y abstractas, así como principios de conservación y esquemas de acción. Por ejemplo, las siguientes fórmulas representan a tres puntos y un triángulo recto:

$$\begin{aligned} p_1 &= punto(t, x_1 : y_1) \\ p_2 &= punto(t, x_2 : y_2) \\ p_3 &= punto(t, x_3 : y_3) \\ tr_1 &= triangulo\_recto(t, p_1, p_2, p_3) \end{aligned} \tag{1.8}$$

Cada igualdad asocia a una constante (en el lado izquierdo) con una descripción geométrica (en el lado derecho); p_1, p_2 y p_3 son constantes de tipo *punto*, y tr_1 es una constante de tipo *triángulo recto*; cada término $punto()$ define a un punto a partir de sus coordenadas en el plano xy, mientras que el término $triangulo\_recto()$ define a un triángulo recto a partir de tres puntos que corresponden a sus vértices. El argumento t de cada término es un valor booleano que resulta verdadero si al evaluar el término se produce una descripción bien definida. Por ejemplo, el argumento t del término $triangulo\_recto()$ resulta verdadero sólo si los tres puntos especificados no son colineales entre sí. En el resto de esta sección, este argumento se omite por claridad.

El lenguaje de representación geométrica define los siguientes elementos:

- Un conjunto de tipos geométricos, como *punto*, *línea*, *triángulo recto*, *cuadrado* y *polígono*.
- Un conjunto de tipos no geométricos, como *booleano*, *real*, y *par real*.
- Un conjunto de símbolos para identificar constantes, y otro para identificar variables, de todos los tipos anteriores.
- Un conjunto de operadores para construir términos geométricos, como $punto()$ y $triangulo\_recto()$.
- Un conjunto de operadores para seleccionar propiedades y relaciones entre términos, por ejemplo, operadores para verificar el tipo de un término, para determinar si dos términos son iguales, para determinar si dos objetos *línea* se interceptan, o para determinar si dos objetos *triángulo recto* están alineados por sus hipotenusas.
- Un operador de abstracción λ, que permite expresar funciones geométricas abstractas. Por ejemplo, la función $\lambda([x, y], alineados\_HH(x, y))$ devuelve *verdadero* para todos los argumentos x, y que representan triángulos rectos alineados por sus hipotenusas.
- Un conjunto de operadores de transformación que permiten modificar objetos geométricos, y que al incluirse en funciones geométricas abstractas permiten definir esquemas de acción. Se incluyen, por ejemplo, transformaciones de traslación y rotación.
- Un operador de descripción geométrica <=, que permite hacer referencia a objetos o configuraciones dentro de un contexto geométrico específico, o a objetos que emergen de la composición de otros objetos geométricos. Por ejemplo, la fórmula $x <= \lambda([x, y], alineados\_HH(x, y))$ permite hacer referencia a un triángulo recto x tal que, en el contexto geométrico definido por $\lambda([x, y], alineados\_HH(x, y))$, está alineado por su hipotenusa con otro triángulo recto y. Este operador permite expresar la reinterpretación de diagramas, como se mostrará al abordar el teorema de Pitágoras.
- Un operador $aplicar()$, que permite expresar la aplicación de funciones de forma explícita. Éste se encarga de aplicar una función o descripción geométrica a una lista de argumentos. Por ejemplo, $aplicar(P, X)$ aplica la función o descripción P a la lista de argumentos X.

Para interpretar este lenguaje se implementó un algoritmo en Prolog que se encarga de evaluar los términos y operadores mencionados.

El teorema de Pitágoras

La Figura 1.27 presenta una demostración diagramática del teorema de Pitágoras. El teorema establece que, para todo triángulo recto con catetos a y b e hipotenusa c, se cumple que $c^2 = a^2 + b^2$, es decir, que el área de un cuadrado construido sobre la hipotenusa (diagrama 4) es igual a la suma de las áreas de los cuadrados construidos sobre los catetos (diagrama 6).

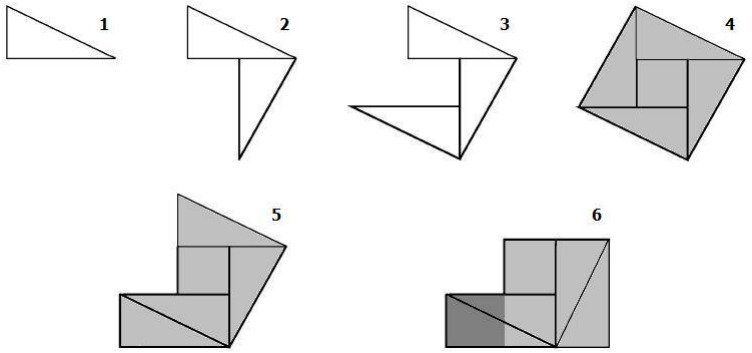


Figura 1.27. Demostración diagramática del teorema de Pitágoras (Pineda, 2007).

En la Figura 1.27, la secuencia inicia con un triángulo recto cualquiera (diagrama 1) y construye un cuadrado sobre su hipotenusa (diagrama 4), para lo cual agrega al diagrama tres triángulos idénticos al primero y un cuadrado con lado $b - a$ (asumiendo que $a \leq b$). Las siguientes dos transformaciones no alteran el área total del diagrama, pues sólo giran al triángulo izquierdo y al superior para alinear sus hipotenusas con los triángulos inferior y derecho, respectivamente (diagrama 6). El área del diagrama 4 es igual a c^2, mientras que el área del diagrama 6 es $a^2 + b^2$. Como las transformaciones entre los diagramas 4 y 6 conservaron el área, entonces se genera el concepto $c^2 = a^2 + b^2$. Esta secuencia diagramática puede realizarse con cualquier triángulo recto inicial, de modo que la secuencia representa a una clase abstracta de configuraciones, y el concepto generado representa un teorema geométrico.

En la presente teoría, la secuencia diagramática de la Figura 1.27 puede generarse a partir de los siguientes elementos:

- Un *principio de conservación de área*, que permite identificar a los esquemas de acción que no alteran el área de un diagrama.
- Un esquema de acción que, dado un triángulo recto, crea otro triángulo igual y lo alinea con el primero, como en el diagrama 2. Como este esquema modifica las figuras del diagrama, representa un *esquema de acción motriz*.
- Un esquema de acción que, dados cuatro triángulos rectos alineados como en el diagrama 4, crea un cuadrado entre ellos con lado $b - a$. Este también es un esquema de acción motriz. Además, como el cuadrado *emerge* de la configuración de los triángulos, este esquema representa una *reinterpretación* del diagrama.
- Un esquema de acción que, dados cuatro triángulos rectos y un cuadrado, alineados como en el diagrama 4, describe al diagrama como "un cuadrado sobre la hipotenusa de un triángulo recto". Este esquema no dibuja nuevas figuras, sino que agrega una descripción al

diagrama, por lo que es un *esquema de acción perceptual*. Como el anterior, también es una reinterpretación del diagrama. Además, satisface el principio de conservación de área.

- Un esquema de acción que, dados dos triángulos rectos alineados como en el diagrama 2, gira a uno de ellos (el *foco de atención*) usando como eje el vértice común de ambos (el *pivote*) de modo que se alineen sus hipotenusas (Figura 1.26). Este también es un esquema de acción motriz y satisface el principio de conservación de área.
- Un esquema de acción que, dados cuatro triángulos rectos y un cuadrado, alineados como en el diagrama 6, describe al diagrama como "dos cuadrados sobre los catetos de un triángulo recto". Este es un esquema de acción perceptual, es una reinterpretación y satisface el principio de conservación de área.

El lenguaje de descripción geométrica definido permite expresar estos elementos, así como la demostración diagramática de la Figura 1.27. Además, el programa prototipo implementado permite evaluar estas expresiones y generar de forma interactiva la secuencia diagramática mostrada.

La suma de los números impares

La presente teoría también permite sintetizar teoremas aritméticos que pueden representarse diagramáticamente. Un ejemplo de lo anterior es un teorema sobre la suma de los números impares, que puede escribirse como $1 + 3 + 5 + 7 + \cdots + (2n - 1) = \sum_{i=1}^{n}(2i - 1) = n^2$. La Figura 1.28 representa este teorema.

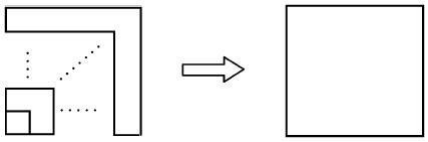


Figura 1.28. Teorema sobre la suma de los números impares (Pineda, 2007).

En esta figura, el área de cada objeto de la izquierda representa a un número impar, mientras que el área del cuadrado de la derecha representa la suma de estos números. El cuadrado pequeño de la izquierda mide 1x1; el *marco* adyacente a éste mide 2x2, y puede verse como tres cuadrados unitarios; el marco siguiente mediría 3x3 y estaría formado por cinco cuadrados unitarios, y así sucesivamente. Por tanto, la figura muestra que la suma de los primeros n numeros impares es igual a n^2.

En la presente teoría, este teorema puede sintetizarse a partir de los siguientes elementos:

- Una definición para el objeto geométrico *marco*, que permite especificar marcos con cualquier tamaño y posición mediante el lenguaje de descripción geométrica.
- Un principio de conservación de área, como el definido para el teorema de Pitágoras.
- Un esquema de acción que, dados un cuadrado con lado l y un marco con lado $l + 1$ adyacente a él, describe al diagrama como "un cuadrado con lado $l + 1$". Este esquema es perceptual, es una reinterpretación y satisface el principio de conservación de área.
- Una función que representa la inducción geométrica sobre cuadrados y marcos, como muestra la Figura 1.28. Esta función permite expresar que el esquema de acción anterior puede aplicarse para cualquier valor de l.

Los primeros tres elementos permiten sintetizar la parte constructiva de la demostración, es decir, que el área de un cuadrado con lado l mas el área de un marco con lado $l + 1$ es igual al área de un cuadrado con lado $l + 1$. Este es el concepto geométrico básico que subyace a la demostración del teorema. El cuarto elemento permite expresar que este concepto geométrico es válido para cualquier valor de l. Este elemento debe definirse de antemano, pues sintetizarlo automáticamente es un problema complejo que se tiene contemplado como trabajo futuro.

El programa prototipo

Para probar la teoría propuesta se implementó un programa prototipo, el cual es capaz de interpretar el lenguaje geométrico definido, aplicar principios de conservación y aplicar esquemas de acción. El programa también es capaz de interpretar un lenguaje aritmético que permite definir correspondencias entre conceptos geométricos y aritméticos. Por ejemplo, para sintetizar el concepto aritmético del teorema de Pitágoras a partir de su demostración diagramática (Figura 1.27), se establece una correspondencia entre la suma de áreas y la suma aritmética, y otra entre la igualdad de áreas y la igualdad aritmética, para poder sintetizar el concepto aritmético $c^2 = a^2 + b^2$.

En este programa prototipo, el proceso de derivación diagramática no es determinista; además, el tamaño del espacio de búsqueda depende del número de principios de conservación y esquemas de acción definidos, y de la forma en que éstos pueden aplicarse. Cada esquema de acción recibe un diagrama y un foco de atención y devuelve un nuevo diagrama; en el caso de los esquemas motrices, también se devuelve una referencia a la figura creada por el esquema, conocida como la *figura paciente*. Por tanto, el estado actual del programa consiste en el diagrama actual, la figura paciente actual y los conceptos geométricos y aritméticos acumulados hasta el momento.

El programa consta de un ciclo principal de resolución, y en cada iteración de este ciclo se aplica un esquema de acción. Cuando el diagrama actual satisface las condiciones de varios esquemas de acción, estos esquemas se presentan al usuario para que decida cuál aplicar. En la síntesis del teorema de Pitágoras, los esquemas aplicables se presentan en orden siguiendo tres heurísticas:

a) Si el diagrama actual contiene al diagrama 4 de la Figura 1.27, sin el cuadrado central, se sugiere el esquema que crea a este cuadrado. Además, los cuatro triángulos alineados se sugieren como el foco de atención del esquema.
b) Si se acaba de agregar un triángulo al diagrama (como en los diagramas 2 y 3 de la Figura 1.27), se sugiere el esquema que agrega un triángulo al diagrama, y se sugiere como foco el triángulo recién agregado (la figura paciente).
c) Si hay dos triángulos alineados como en el diagrama 2 de la Figura 1.27, pero ninguno de ellos se acaba de agregar al diagrama (ninguno es la figura paciente), se sugiere el esquema que alinea las hipotenusas de los dos triángulos, y se sugiere como foco cualquiera de los dos triángulos alineados.

Al evaluar estas heurísticas se consideran todas las combinaciones de triángulos posibles en el diagrama actual. Además, no se permite aplicar esquemas que producirían el traslape de figuras. El programa también le permite al usuario deshacer el último esquema aplicado, de modo que pueda explorar el espacio de búsqueda.

El usuario también debe decidir cuándo aplicar los principios de conservación disponibles, es decir, cuándo restringir la lista de esquemas de acción a aquellos que cumplen ciertos principios de conservación (como en el caso de los diagramas 4 a 6 de la Figura 1.27).

El programa también es capaz de sintetizar el teorema sobre la suma de los números impares siguiendo el procedimiento interactivo descrito, pero utilizando como figura inicial un cuadrado y un marco alineados, en lugar de un triángulo recto. Para realizar esta síntesis se deben incluir en el programa los principios de conservación y esquemas de acción necesarios, como se describió anteriormente en *La suma de los números impares*.

En el futuro se planea extender este programa con una interfaz gráfica interactiva, que ayude al usuario a visualizar configuraciones potencialmente interesantes en el proceso de derivación diagramática. En el prototipo actual, el usuario debe examinar el diagrama actual y decidir qué acciones realizar. La síntesis automática de conceptos geométricos es una tarea compleja que requiere una formalización de los procesos de inferencia perceptual involucrados.

Conclusiones

En este trabajo se presentó una teoría para la síntesis de algunos conceptos geométricos y aritméticos. Esta teoría está basada en principios de conservación y esquemas de acción, los cuales pueden considerarse como formas particulares de razonamiento esquemático. La teoría también podría ser efectiva en la síntesis de conceptos lógicos, lo cual se contempla como trabajo futuro.

Los conceptos sintetizados mediante esta teoría podrían utilizarse en otros procesos de inferencia, como deducción, abducción, simulación y satisfacción de restricciones. La interacción entre estos modos de razonamiento representa un tema interesante para futuras investigaciones.

Los elementos de representación e inferencia presentados permiten analizar las propiedades de las representaciones diagramáticas en general, por ejemplo:

- Cuál es el poder expresivo de los diagramas.
- Por qué los diagramas son efectivos para la comunicación y la inferencia.
- Cuál es la relación entre las inferencias sintéticas (involucradas en la generación de conceptos) y las inferencias válidas (involucradas en la demostración de los mismos).

Para un análisis más detallado de esta teoría véase Pineda (2007).

1.4 Objetivos de este proyecto

El objetivo de este trabajo es implementar un programa que resuelva la RPM estándar automáticamente, tomando como base la teoría y el programa prototipo propuestos por Pineda (2007) y descritos en la Sección 1.3.

Debido a límites de tiempo no se abordarán mecanismos para manipular imágenes sin procesar (representadas con pixeles), sino que se partirá de una representación simbólica de la prueba. Esta representación será elaborada manualmente, como en la mayoría de los programas presentados en

la Sección 1.2. Sin embargo, el programa podrá extenderse en el futuro para procesar imágenes directamente.

El programa se centrará en modelar la inducción de reglas, por las siguientes razones:

- La RPM busca medir la *habilidad eductiva*, es decir, la habilidad para descubrir relaciones a partir de información nueva. Por tanto, el proceso crucial en su resolución es la inducción de reglas y no la aplicación de reglas predefinidas.
- De los nueve programas que se han implementado para resolver la RPM, sólo tres modelan la inducción de reglas y su desempeño es deficiente.
- La teoría propuesta por Pineda (2007) permite sintetizar conceptos geométricos, por lo que es interesante investigar si esta teoría puede aplicarse para sintetizar las reglas requeridas para resolver la RPM.

En suma, este proyecto permitirá investigar los siguientes temas:

- El proceso de inducción de reglas en la resolución de la RPM, una prueba de *habilidad eductiva* importante en el estudio de la inteligencia.
- La aplicabilidad de la teoría propuesta por Pineda (2007) para resolver problemas de razonamiento abstracto como los de la RPM.
- Los procesos involucrados en la inteligencia, como la habilidad para realizar abstracciones, inducir relaciones y realizar analogías.

1.5 Resumen del capítulo

La prueba de matrices progresivas de Raven (RPM) es una prueba de *habilidad eductiva* ampliamente utilizada e investigada, que está asociada al *factor g* de la teoría de la inteligencia de Charles Spearman. Aunque su validez para "medir la inteligencia" es un tema polémico, es una herramienta útil para investigar la inteligencia.

Se han implementado nueve programas para resolver la RPM de forma automática. Aunque éstos difieren en su forma de representar los problemas de la prueba, en su modelación de la inducción de reglas, en su modelación del aprendizaje y en los resultados obtenidos, todos resaltan la importancia del razonamiento por analogía y presentan conclusiones relevantes con respecto a la RPM, la inteligencia humana y la inteligencia artificial.

El programa que se desarrollará en este proyecto está basado en la teoría de Pineda (2007) para la síntesis de conceptos geométricos. Esta teoría involucra principios de conservación, esquemas de acción, la descripción de abstracciones geométricas y la reinterpretación de diagramas. La teoría se aplicó para sintetizar el teorema de Pitágoras y un teorema sobre la suma de los números impares; además se implementó en un programa prototipo interactivo que asiste al usuario en la síntesis de conceptos geométricos.

En el presente trabajo se retomará esta teoría para desarrollar un programa que resuelva la RPM estándar automáticamente. El programa partirá de una representación simbólica de la prueba elaborada a mano, pero podrá extenderse en el futuro para manipular imágenes directamente. El programa se centrará en modelar el proceso de inducción de reglas, un elemento crucial en la resolución de la prueba que no ha sido suficientemente estudiado en trabajos anteriores.

Este proyecto permitirá investigar la aplicabilidad de la teoría propuesta por Pineda (2007) para resolver problemas de razonamiento abstracto. Además permitirá analizar las características de la RPM, los procesos necesarios para su resolución y los procesos involucrados en la inteligencia.

2 Resolución automática de la prueba

Este capítulo describe el modelo desarrollado en este proyecto para resolver la RPM estándar, así como sus resultados. El modelo tomó como base la teoría propuesta por Pineda (2007) para la síntesis de conceptos geométricos, descrita en la Sección 1.3.

La Sección 2.1 presenta el lenguaje utilizado para representar simbólicamente cada problema de la prueba. La Sección 2.2 describe cómo se inducen reglas a partir de esta representación. La Sección 2.3 explica cómo se seleccionan las reglas inducidas para identificar a las más relevantes. La Sección 2.4 muestra cómo se utilizan las reglas seleccionadas para elegir la respuesta de un problema. La Sección 2.5 expone los resultados del modelo en la RPM estándar. La Sección 2.6 expone los resultados del modelo en problemas fabricados más generales. Por último, la Sección 2.7 presenta un resumen del capítulo.

2.1 Lenguaje de representación geométrica

El primer paso para resolver un problema de la RPM es identificar a las figuras que lo componen, lo cual no es un proceso trivial. Esto se debe a que un mismo diagrama puede interpretarse de diferentes formas, como muestra la Figura 2.1.

Figura 2.1. Diferentes interpretaciones de un diagrama.

El diagrama de la izquierda puede interpretarse como una flecha hacia arriba, o como un triángulo con una línea vertical debajo. Sin embargo, al comparar el diagrama de la izquierda con el del centro, el de la izquierda puede reinterpretarse como una T con un ángulo encima, como muestra el diagrama de la derecha. Esta última interpretación produce una descripción de ambos diagramas que es más eficiente que una basada en pixeles o en líneas individuales, pues sólo requiere almacenar dos clases de figuras en dos posiciones específicas: un ángulo arriba y una T abajo. Una descripción con pixeles sólo requiere una clase de figura (el pixel), pero requiere cientos o miles de posiciones diferentes. Una descripción con líneas, en este caso, requiere una clase de figura (la línea) en cuatro posiciones y escalas diferentes.

En el contexto de la teoría de Pineda, al comparar dos imágenes se realiza un esquema de acción perceptual que genera una descripción simbólica y eficiente de las imágenes, en términos de sus diferencias y similitudes. Este esquema perceptual se utilizará como base para representar simbólicamente los problemas de la RPM.

Representación de un problema

En los problemas de la prueba, una misma figura puede cambiar de posición, orientación o tamaño en diferentes celdas de la matriz, como muestra la Figura 2.2.

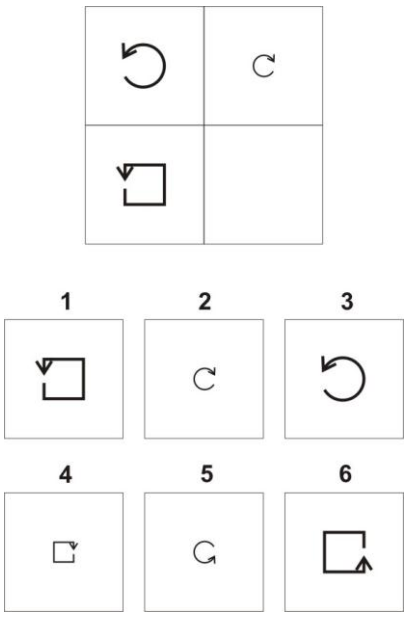


Figura 2.2. Transformación de figuras. La respuesta es 4.

Al comparar las celdas de la matriz y las respuestas entre sí, es posible describir este problema utilizando sólo dos clases de figuras: la flecha curva y la flecha cuadrada. Además, cada clase de figura aparece en dos tamaños y tres orientaciones diferentes. En el lenguaje de representación propuesto, la celda superior izquierda de la matriz se representa de la siguiente forma:

```
diagrama([
      figura([ [nombre, flechaCurva], [posición,50:50] ])
])
```

Esta sintaxis está basada en la del lenguaje Prolog, en la que los corchetes cuadrados se utilizan para representar listas. La cláusula `diagrama()` se utiliza para representar una celda de la matriz o una respuesta del problema, y contiene una lista de figuras. La cláusula `figura()` representa a una figura del problema, y contiene una lista de pares atributo-valor. Los atributos considerados son:

- `nombre`: es una etiqueta arbitraria que indica la clase de la figura, y no puede omitirse.
- `posición`: es una pareja de enteros X:Y que indica la posición del centro de la figura dentro de un diagrama. El centro de una figura se define como el promedio de las coordenadas máximas y mínimas de la figura en cada eje. La ubicación del origen del plano coordenado en el diagrama es arbitraria, pero debe ser la misma en todos los diagramas de un mismo problema. Si el atributo `posición` se omite, se asume que tiene un valor 0:0.
- `escalar`: es una pareja de números reales S_x:S_y mayores que cero, que indica si la figura fue escalada horizontal o verticalmente. Un valor 0.5:1 indica que la figura fue escalada al 50% en el eje X y que no fue escalada en el eje Y. Su valor por defecto es 1:1.

- `rotar`: es un número real que indica si la figura fue rotada. La rotación se define en grados y en contra del sentido del reloj. El entero puede ser positivo o negativo, pero el intérprete del lenguaje representará este valor en el intervalo [0,360). Su valor por defecto es 0.
- `reflejarX`: es un valor verdadero/falso que indica si el eje X de la figura fue invertido, es decir, si su lado izquierdo pasó a su lado derecho. El reflejo del eje Y se obtiene combinando un reflejo en X con una rotación de 180°. Su valor por defecto es `falso`.

Los atributos `escalar`, `rotar` y `reflejarX` de una figura se definen con respecto a una figura cualquiera de su misma clase: la figura representante. En el ejemplo de la Figura 2.2 se tomarán como representantes a las figuras de la columna izquierda de la matriz.

Un problema completo se representa de la siguiente forma:

```
problema(
      nombreDelProblema,
      matriz( númeroDeFilas, [...]),
      respuestas( [...] ),
      solución( [...] )
).
```

Donde `nombreDelProblema` es una etiqueta arbitraria para identificar al problema. La cláusula `matriz()` representa a la matriz del problema. Su primer argumento es un entero que indica el número de filas en la matriz, y su segundo argumento es una lista de diagramas (uno para cada celda de la matriz). Las celdas de la matriz se listan de izquierda a derecha y de arriba a abajo, y el número de diagramas en la lista debe ser un múltiplo de `númeroDeFilas`. La o las celdas faltantes de la matriz se representan con la cláusula `diagrama([nulo])`.

La cláusula `respuestas()` contiene una lista de diagramas (uno para cada respuesta del problema). La cláusula `solución()` contiene la respuesta correcta del problema, y sólo se utiliza para evaluar el desempeño del programa de forma automática; ésta contiene una lista de enteros en el intervalo [1,*n*], donde *n* es el número de respuestas del problema, y permite considerar problemas en los que la matriz contiene varias celdas faltantes. Las respuestas de la cláusula `solución()` se asignan a las celdas faltantes de la matriz de izquierda a derecha y de arriba a abajo. Por ejemplo, si a una matriz de 2x2 le faltan la celda superior izquierda y la inferior derecha, la solución [3,1] asigna la tercera respuesta a la celda superior izquierda y la primera respuesta a la celda inferior derecha.

En suma, el problema de la Figura 2.2 se representa de la siguiente forma:

```
problema( problemaEjemplo,
      matriz(2, [
            diagrama([
                  figura([ [nombre,flechaCurva], [posición,50:50] ])
            ]),
            diagrama([
                  figura([ [nombre,flechaCurva], [posición,50:50],
                        [escalar,0.5:0.5], [reflejarX] ])
            ]),

            diagrama([
                  figura([[nombre,flechaCuadrada],[posición,50:50]])
```

```
                ]),
                diagrama([nulo])
        ]),
        respuestas([
                diagrama([
                        figura([[nombre,flechaCuadrada],[posición,50:50]])
                ]),
                diagrama([
                        figura([ [nombre,flechaCurva], [posición,50:50],
                                [escalar,0.5:0.5], [reflejarX] ])
                ]),
                diagrama([
                        figura([ [nombre,flechaCurva], [posición,50:50] ])
                ]),

                diagrama([
                        figura([ [nombre,flechaCuadrada],
                                [posición,50:50], [escalar,0.5:0.5],
                                [reflejarX] ])
                ]),
                diagrama([
                        figura([ [nombre,flechaCurva], [posición,50:50],
                                [escalar,0.5:0.5], [rotar,180] ])
                ]),
                diagrama([
                        figura([ [nombre,flechaCuadrada],
                                [posición,50:50], [rotar,180] ])
                ])
        ]),
        solución([4])
).
```

Relleno de las figuras

En algunos problemas una figura puede aparecer con rellenos diferentes, como en la Figura 2.3.

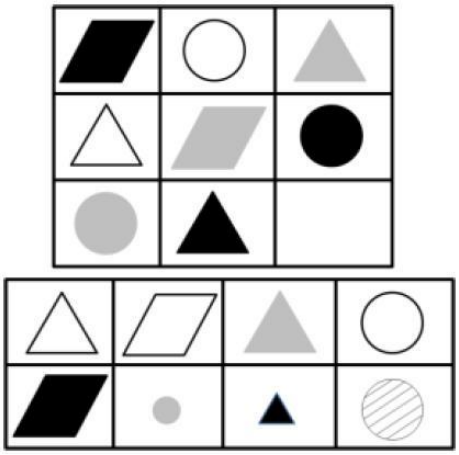


Figura 2.3. Figuras con relleno diferente (Lovett et al., 2010). La respuesta es el cuadrilátero sin relleno.

Para este problema es necesario reconocer que algunas figuras tienen la misma forma, y que otras tienen el mismo relleno. En el lenguaje propuesto se optó por representar al relleno como una figura más en un diagrama, de modo que el relleno sea independiente de las figuras y pueda tener sus propios valores de posición, orientación y escala sin agregar complejidad al lenguaje. De este modo, la celda superior izquierda de la Figura 2.3 se representa como:

```
diagrama([
      figura([ [nombre, cuadrilátero], [posición,50:50] ]),
      figura([ [nombre, rellenoNegro], [posición,50:50] ])
])
```

Mientras que la celda inferior central se representa como:

```
diagrama([
      figura([ [nombre, triángulo], [posición,50:50] ]),
      figura([ [nombre, rellenoNegro], [posición,50:50] ])
])
```

Aunque no tiene sentido trasladar, escalar, rotar o reflejar un relleno uniforme, estos atributos son útiles en rellenos no uniformes como los rayados o cuadriculados. El atributo `posición` puede determinarse con base en cualquier punto característico del relleno, pero debe ser consistente en todos los diagramas de un problema.

Problemas con una sola imagen

En problemas como el de la Figura 2.4, la matriz es en realidad una sola imagen. Sin embargo, este problema puede representarse con una matriz de 2x2 como muestra la Figura 2.5.

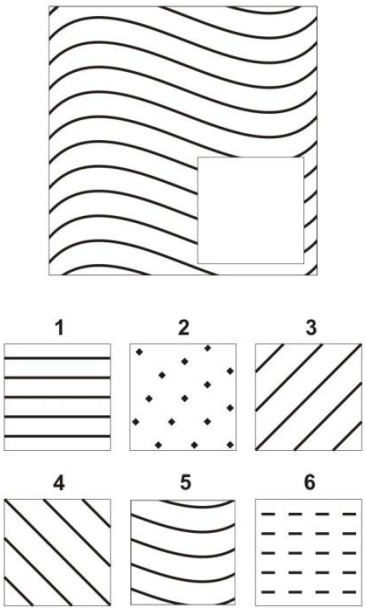
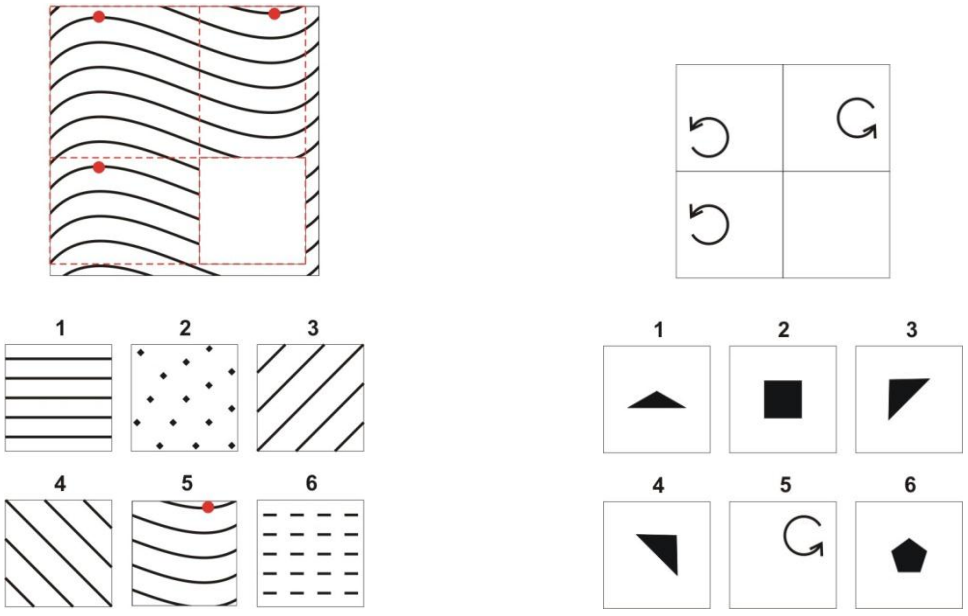


Figura 2.4. Problema con una sola imagen.



a) b)

Figura 2.5. Representación de un problema con una sola imagen. a) Problema original. b) Problema equivalente.

La imagen del problema se divide como muestran las líneas discontinuas de la Figura 2.5a y se descarta el borde sobrante. Al comparar las celdas creadas y las respuestas entre sí, se observa que las tres celdas y la respuesta 5 contienen el mismo relleno en diferentes posiciones y orientaciones, mientras que las demás respuestas contienen rellenos diferentes. Como se mencionó anteriormente, la posición de un relleno se determina en referencia a cualquier punto característico del relleno, pero este punto debe ser consistente en todo el problema. Por ejemplo, al usar los centros de los círculos rojos de la Figura 2.5a como puntos de referencia, el relleno de la celda superior izquierda es igual al de la celda superior derecha desplazado hacia la izquierda y hacia abajo, y rotado 180°. Cada relleno se representa como una sola figura aunque esté formado por varias líneas o puntos, de modo que la Figura 2.5b representa un problema equivalente al de la Figura 2.5a, en el cual las diferentes posiciones de los rellenos de la matriz se representan exageradas por claridad.

Ventajas y desventajas

El modelo de representación propuesto presenta las siguientes ventajas:

- Establece las bases de un método para crear representaciones simbólicas a partir de información subsimbólica, utilizando un esquema perceptual que identifica las diferencias y similitudes entre conjuntos de imágenes.
- Es general, pues no define un conjunto finito de figuras básicas. El atributo `nombre` es sólo una etiqueta arbitraria que permite distinguir entre figuras iguales y figuras diferentes.
- Permite separar el procesamiento simbólico del procesamiento de imágenes, ya que el atributo `nombre` no proporciona información sobre la forma ni el tamaño de las figuras, y los demás atributos sólo permiten deducir las diferencias relativas entre figuras de la misma clase. Por tanto, el lenguaje omite la información visual del problema (cuáles figuras lo componen), pero conserva la información espacial y relacional del mismo (cómo están ubicadas las figuras, y cuáles figuras son iguales o diferentes).

- Es eficiente, pues permite representar diagramas utilizando pocos elementos. Esto simplifica el procesamiento simbólico posterior, ya que permite representar figuras compuestas como una sola figura.
- Permite representar de un modo simple los rellenos de figuras y sus propiedades.

La principal desventaja de este lenguaje es que el proceso para representar simbólicamente un problema de la RPM es más complejo que reconocer un conjunto finito de figuras básicas. Al comparar las celdas y las respuestas entre sí se genera un conjunto de clases candidatas para representar al problema, pero la selección de las mejores clases es un problema no trivial que se contempla como trabajo futuro. En este proyecto, los problemas se codificaron manualmente siguiendo una heurística de simplicidad: utilizar el menor número posible de clases y figuras.

Al codificar manualmente los problemas se corre el riesgo de influir en los resultados del modelo, dado que la persona que realiza la codificación puede conocer o identificar las respuestas correctas de los problemas. En esta cuestión hay que considerar los siguientes puntos:

- Este riesgo está presente en siete de los nueve programas desarrollados en trabajos previos, los cuales partieron de representaciones simbólicas y hechas a mano de los problemas.
- En estos siete programas y en el presente trabajo, la codificación manual permite modelar el razonamiento simbólico en la prueba evitando problemas propios del procesamiento de imágenes.
- En varios de los problemas de la prueba, la representación propuesta en este trabajo resulta similar a la utilizada en trabajos previos. Por ejemplo, al comparar las celdas y las respuestas del problema de la Figura 2.3 resultan tres clases de figuras: cuadrilátero, triángulo y círculo, las cuales son similares a las figuras básicas utilizadas en trabajos anteriores. En este trabajo, sin embargo, las clases inducidas sólo se utilizan como etiquetas, y se propone un mecanismo mediante el cual pueden inducirse estas clases a partir de imágenes sin procesar.
- Como se verá en la Sección 2.5, los resultados obtenidos por el presente modelo dependen principalmente del mecanismo de inducción y selección de reglas, y no de la codificación utilizada para representar los problemas.

Aunque el tema central de este trabajo no es el procesamiento de imágenes, el modelo de procesamiento simbólico que se presentará a continuación puede contribuir a definir las mejores clases para representar un problema. Por ejemplo, una persona puede seleccionar un conjunto de clases para representar un problema, pero luego descubrir que el problema no presenta una lógica simple con esta representación, y entonces decide que debe reinterpretar el problema en el nivel subsimbólico. De este modo, los niveles simbólico y subsimbólico pueden interactuar para representar y resolver un problema.

2.2 Inducción de reglas

Aunque el lenguaje de representación propuesto simplifica de forma importante los problemas de la prueba, es sólo el primer paso para descubrir reglas como las descritas en la Sección 1.2, por ejemplo, “distribución de valores” o “suma o resta de figuras”. Esta sección describe un modelo que permite generar reglas suficientes para resolver la RPM estándar y otros problemas más generales.

Reglas básicas

En la teoría de Pineda (2007) se genera un espacio de búsqueda de diagramas, a partir de una figura inicial y un conjunto de esquemas de acción, y es posible sintetizar conceptos geométricos abstractos aplicando principios de conservación en este espacio.

En el presente trabajo se genera un espacio de búsqueda de reglas, a partir de las figuras de un problema, un esquema de acción perceptual y un modelo de composición de reglas, y es posible resolver los problemas de la RPM estándar mediante las reglas generadas. El esquema utilizado es la percepción de igualdad, y se ilustra en la Figura 2.6.

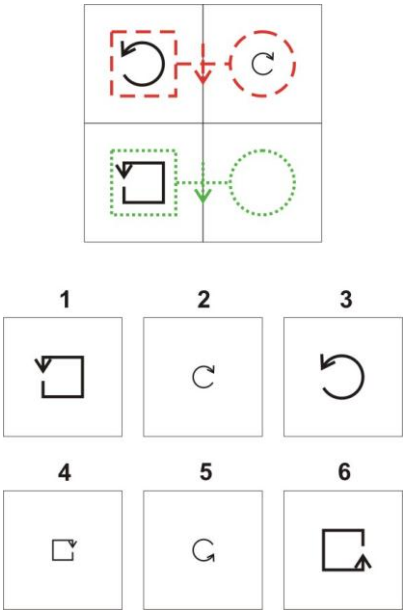


Figura 2.6. Esquema de percepción de igualdad, y regla básica que se extiende por filas.

En esta imagen, las líneas discontinuas rojas representan a la igualdad $f_1 = T(f_2)$, donde f_1 es la flecha curva grande, f_2 es la flecha curva pequeña, y T es la transformación "escalar al 200% y reflejar el eje X". El cuadrado indica que f_1 es el *resultado* de la operación, y el círculo indica que f_2 es un *operando*. Si el cuadrado estuviera en f_2 y el círculo en f_1, las líneas rojas representarían una igualdad inversa a la anterior, es decir, $f_2 = T'(f_1)$, donde T' sería la transformación "escalar al 50% y reflejar el eje X".

A partir de la percepción de igualdad $f_1 = T(f_2)$ se induce una regla básica: $c_{k,0} = T(c_{k,1})$, donde k es un entero mayor o igual que cero, $c_{k,0}$ es una celda en la fila k y en la columna 0, y $c_{k,1}$ es una celda en la fila k y en la columna 1. Esta regla representa una descripción de los diagramas de la matriz, y significa que para cualquier fila, la celda izquierda es el resultado de escalar al 200% y reflejar el eje X de la celda derecha. En la Figura 2.6, las flechas hacia abajo indican que la regla se extiende por filas, de modo que las líneas discontinuas rojas y las líneas punteadas verdes representan dos instancias de la regla $c_{k,0} = T(c_{k,1})$ (con $k = 0$ y $k = 1$, respectivamente).

Si se aplica el esquema perceptual a todas las figuras en celdas diferentes de la matriz, se genera un conjunto de reglas básicas. En el ejemplo de la Figura 2.6 se generan cuatro reglas:

1. $c_{k,0} = T(c_{k,1})$, representada con las líneas discontinuas rojas y las líneas punteadas verdes en la Figura 2.6.
2. $c_{k,1} = T'(c_{k,0})$, o la regla inversa de la anterior.
3. $c_{0,k} = T(c_{0,k+1})$, representada con las líneas discontinuas rojas y las líneas punteadas verdes en la Figura 2.7, y que resulta de extender a la igualdad $f_1 = T(f_2)$ por columnas.
4. $c_{0,k+1} = T'(c_{0,k})$, o la regla inversa de la anterior.

En la Figura 2.7 se observa que la matriz se trata como una estructura toroidal, es decir, se asume que los extremos izquierdo y derecho son contiguos, así como los extremos superior e inferior. De este modo, las líneas discontinuas rojas de la Figura 2.7 representan a la regla $c_{0,0} = T(c_{0,1})$, mientras que las líneas punteadas verdes representan a la regla $c_{0,1} = T(c_{0,2}) = T(c_{0,0})$ (ambas instancias de la regla básica 3, con $k = 0$ y $k = 1$, respectivamente). Aunque este manejo de la matriz puede generar reglas erróneas, se verá que es necesario para poder resolver ciertos tipos de problemas.

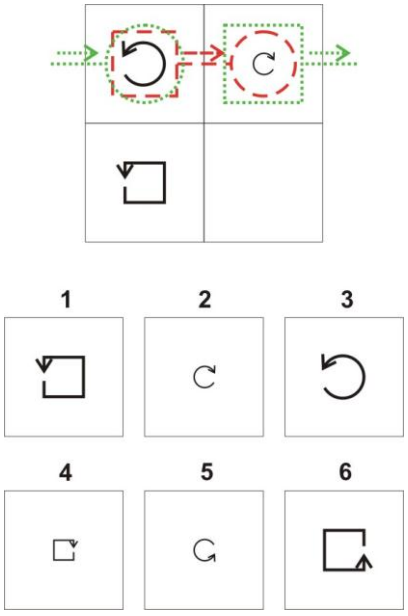


Figura 2.7. Regla básica que se extiende por columnas.

En los teoremas sintetizados por la teoría de Pineda, el espacio de búsqueda es generado por esquemas tanto motrices como perceptuales, los cuales generan nuevos diagramas y nuevas descripciones, respectivamente. En la resolución de la RPM no se busca generar nuevos diagramas, sino inducir nuevas descripciones (reglas) que representen la lógica de los diagramas existentes. Es por esto que los esquemas perceptuales son centrales en este proceso.

En la Sección 2.5 se verá que más de la mitad de la RPM puede resolverse utilizando sólo reglas básicas, pero que el resto de los problemas requiere de reglas compuestas.

Reglas compuestas

La Figura 2.8 presenta un problema que no puede resolverse utilizando sólo reglas básicas.

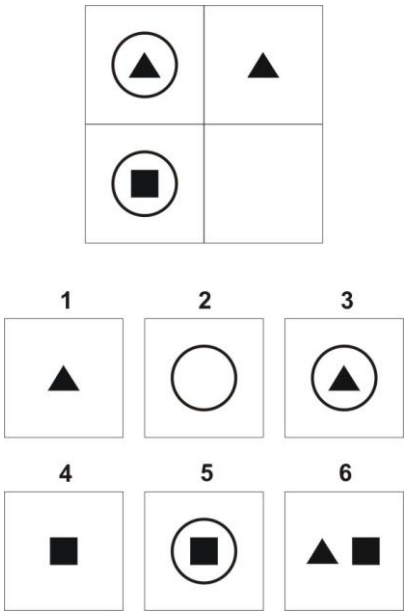


Figura 2.8. Problema que requiere reglas compuestas.

Al aplicar el esquema perceptual a todas las figuras en celdas diferentes de la matriz, se generan ocho reglas básicas, representadas en la Figura 2.9. Las líneas discontinuas rojas representan cuatro de estas reglas, inducidas a partir de la igualdad entre los triángulos de la fila superior de la matriz:

1. $c_{k,0} = T(c_{k,1})$, equivalente a extender las líneas rojas por filas.
2. $c_{k,1} = T'(c_{k,0})$, o la inversa de la anterior.
3. $c_{0,k} = T(c_{0,k+1})$, equivalente a extender las líneas rojas por columnas.
4. $c_{0,k+1} = T'(c_{0,k})$, o la inversa de la anterior.

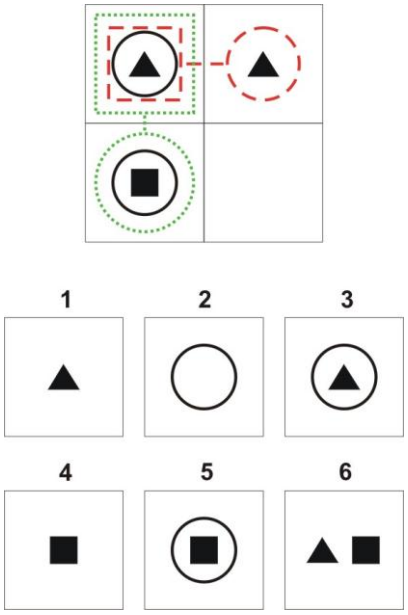


Figura 2.9. Reglas básicas generadas.

En este caso las figuras son idénticas, por lo que la transformación T y su inversa son transformaciones identidad, es decir, no modifican a las figuras. Las líneas punteadas verdes de la

Figura 2.9 representan otras cuatro reglas, inducidas a partir de la igualdad entre los círculos de la columna izquierda de la matriz:

5. $c_{k,0} = T(c_{k+1,0})$, equivalente a extender las líneas verdes por filas.
6. $c_{k+1,0} = T'(c_{k,0})$, o la inversa de la anterior.
7. $c_{0,k} = T(c_{1,k})$, equivalente a extender las líneas verdes por columnas.
8. $c_{1,k} = T'(c_{0,k})$, o la inversa de la anterior.

Estas ocho reglas describen la noción de que la celda faltante debe tener figuras en común con la celda que está arriba de ella y con la celda que está a su izquierda. Sin embargo, no proporcionan información sobre qué figuras deben satisfacer cada regla, de modo que las cinco primeras respuestas satisfacen esta noción de algún modo. Este problema puede resolverse con una regla compuesta, como la representada en la Figura 2.10.

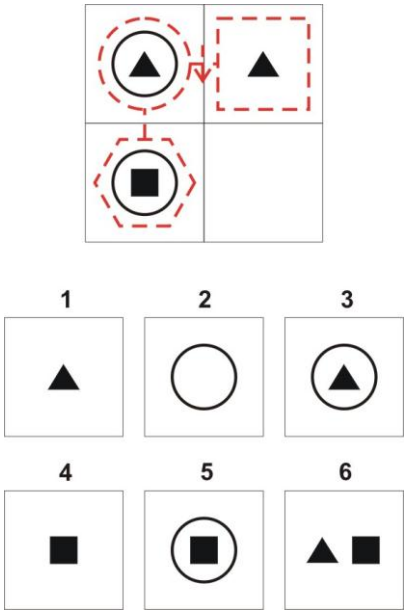


Figura 2.10. Regla compuesta que se extiende por filas.

En la Figura 2.10, las líneas discontinuas rojas representan a la regla $c_{k,1} = T(c_{k,0}) \cap T(c_{k+1,0})^C$, donde $\cap$ y ^{C} son los operadores de intersección y complemento de conjuntos, respectivamente. El cuadrado rojo representa al resultado de la operación ($c_{k,1}$), el círculo rojo a un *operando positivo* ($T(c_{k,0})$), y el hexágono rojo a un *operando negativo* ($T(c_{k+1,0})^C$). La flecha hacia abajo indica que la regla se extiende por filas.

Cuando $k = 0$, la regla describe que la celda superior derecha es igual al conjunto de las figuras que están en la celda superior izquierda pero no están en la celda inferior izquierda. Cuando $k = 1$, la regla describe que la celda inferior derecha es igual al conjunto de las figuras que están en la celda inferior izquierda pero no están en la celda superior izquierda, lo cual produce la respuesta correcta del problema.

La regla de la Figura 2.10 relaciona a tres celdas, por lo que se considera una regla de nivel tres. Las reglas básicas sólo relacionan a dos celdas, por lo que se consideran reglas de nivel dos.

Construyendo reglas compuestas

Mientras que las reglas básicas se generan a partir de las figuras del problema, mediante el esquema de percepción de igualdad, las reglas compuestas se generan combinando reglas del nivel inmediato inferior con reglas básicas. Por ejemplo, la regla compuesta de la Figura 2.10 se genera combinando las dos reglas básicas representadas en la Figura 2.11.

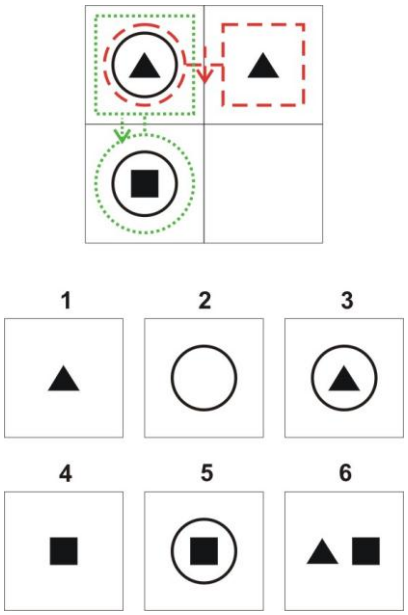


Figura 2.11. Composición de dos reglas básicas que se extienden por filas.

En la Figura 2.11, las líneas discontinuas rojas representan a la regla $c_{k,1} = T_1(c_{k,0})$, mientras que las líneas punteadas verdes representan a la regla $c_{k,0} = T_2(c_{k+1,0})$, donde T_1 y T_2 son transformaciones identidad y ambas reglas se extienden por filas. La regla compuesta se construye observando que el operando de la primera regla (el círculo rojo) es también el resultado de la segunda regla (el cuadrado verde). Esto indica que el círculo rojo tiene figuras en común con el círculo verde, de modo que, al agregar al círculo verde como un operando negativo en la regla roja, el resultado de la nueva regla tendrá menos figuras que el resultado de la regla roja. En otras palabras, la nueva regla será un refinamiento de la regla original.

La transformación T_1 de la regla roja se aplica a la transformación T_2 de la regla verde, para que todas las figuras involucradas se alineen con el resultado de la regla roja. En este caso, el círculo verde se está agregando como un operando negativo a la regla roja, de modo que la regla compuesta resultante es $c_{k,1} = T_1(c_{k,0}) \cap T_1(T_2(c_{k+1,0}))^C$; T_1 y T_2 son transformaciones identidad, por lo que la regla compuesta equivale a $c_{k,1} = T_1(c_{k,0}) \cap T_1(c_{k+1,0})^C$.

Las reglas compuestas también podrían construirse agregando todos los operandos posibles a las reglas del nivel inmediato inferior, sin considerar a las reglas básicas, pero esto causaría varios inconvenientes:

- Las reglas generadas sólo dependerían del tamaño de la matriz y no de las figuras que contiene. Para todas las matrices del mismo tamaño, siempre se generarían las mismas reglas y su número sería exponencial.

- Al ignorar el contenido de la matriz para generar las reglas, se pasaría de inducir reglas a producir reglas. El razonamiento inductivo consiste en obtener conclusiones generales a partir de casos particulares, no en generar conclusiones indiscriminadamente.
- Muchas de las reglas construidas serían irrelevantes, pues se agregarían operandos que no afectan el resultado de la regla original, o bien que producen como resultado un diagrama vacío.

En la construcción de reglas compuestas se siguen las siguientes heurísticas:

- Sólo se combinan reglas del mismo tipo (reglas por filas con reglas por filas, y reglas por columnas con reglas por columnas). Esto permite que las reglas conserven una misma estructura conforme se extienden.
- Al agregar un operando negativo a una regla, este operando debe tener figuras en común con alguno de los operandos originales de la regla. Es decir, debe existir una regla básica que relacione al nuevo operando con alguno de los operandos originales. Esto permite que la nueva regla sea un refinamiento de la regla original, pues su resultado contendrá menos figuras.
- Al agregar un operando positivo a una regla, este operando debe tener figuras en común con el resultado de la regla original. Es decir, debe existir una regla básica que relacione al nuevo operando con el resultado de la regla. De no ser así, al calcular la intersección de los operandos se produciría un diagrama que no tendría figuras en común con el diagrama correspondiente al resultado de la regla.
- Las reglas de nivel k se construyen combinando reglas de nivel $k-1$ con reglas básicas, sin embargo, no se utilizan todas las reglas de nivel $k-1$ para construir nuevas reglas. Sólo se utiliza un conjunto de reglas relevantes de nivel $k-1$, lo cual permite refinar sólo a las mejores reglas y reducir considerablemente el número de reglas generadas.

La utilidad de estas heurísticas se apreciará mejor en la Sección 2.3, al abordar qué distingue a las reglas relevantes de las irrelevantes.

Espacio de búsqueda de reglas

El esquema de percepción de igualdad propuesto, junto con el modelo de composición de reglas, producen un espacio de búsqueda de reglas. La estructura de estas reglas está basada en la forma normal disyuntiva utilizada en lógica proposicional:

$$(x_1 \wedge x_2 \wedge \ldots \wedge x_m) \vee (y_1 \wedge y_2 \wedge \ldots \wedge y_n) \vee \ldots \vee (z_1 \wedge z_2 \wedge \ldots \wedge z_p)$$

Cualquier fórmula proposicional puede expresarse en esta forma. En particular, cualquier fórmula proposicional construida con los operadores de conjunción ($\wedge$), disyunción ($\vee$) y negación ($\neg$) puede convertirse a la forma normal disyuntiva mediante las siguientes equivalencias lógicas:

- $\neg(\neg x) = x$, o eliminación de la doble negación.
- $\neg(x \wedge y) = \neg x \vee \neg y$, $\neg(x \vee y) = \neg x \wedge \neg y$, o las leyes de De Morgan.
- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$, o las leyes distributivas.

Como los operadores de intersección ($\cap$), unión ($\cup$) y complemento (^{c}) en teoría de conjuntos también satisfacen estas equivalencias, entonces cualquier fórmula entre conjuntos construida con estos operadores también puede expresarse en la forma normal disyuntiva.

Cada regla representa de forma explícita una intersección de operandos positivos y negativos, mientras que la unión se representa de forma implícita mediante la superposición de reglas, como muestra la Figura 2.12.

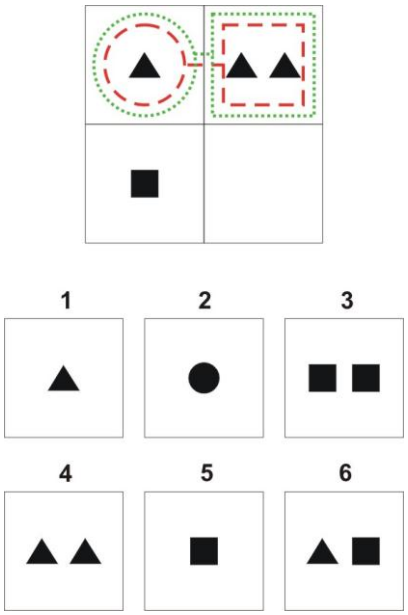


Figura 2.12. Superposición de reglas.

En esta figura, las líneas discontinuas rojas representan a la regla $c_{0,1} = T_1(c_{0,0})$, donde T_1 es un desplazamiento hacia la izquierda. Las líneas punteadas verdes representan a la regla $c_{0,1} = T_2(c_{0,0})$, donde T_2 es un desplazamiento hacia la derecha. Al aplicar ambas reglas de forma simultánea se produce la regla $c_{0,1} = T_1(c_{0,0}) \cup T_2(c_{0,0})$, que indica que la celda superior derecha es igual a dos copias de la celda superior izquierda, con una copia desplazada hacia la izquierda y otra hacia la derecha.

Por lo tanto, el espacio de búsqueda de reglas definido permite expresar cualquier fórmula entre conjuntos construida con los operadores de intersección, unión y complemento.

Programación lógica inductiva

El modelo de composición de reglas está basado en la técnica de programación lógica inductiva (ILP), la cual permite inducir relaciones a partir de ejemplos[13]. Un problema de ILP consta de los siguientes elementos:

- Un conjunto de ejemplos positivos y un conjunto de ejemplos negativos.
- Una base de conocimiento, tal que los ejemplos positivos no puedan inferirse a partir de ésta.

[13] Para una introducción a la programación lógica inductiva, véase Bratko (2001), capítulo 19.

Las relaciones a inducir deben satisfacer todos los ejemplos positivos pero ningún ejemplo negativo, es decir, deben ser *completas* y *consistentes*. En el contexto de la RPM, las figuras presentes o ausentes en cada celda de la matriz representan ejemplos positivos y negativos, respectivamente, y las reglas a inducir deben satisfacer a las figuras presentes pero no a las ausentes.

La base de conocimiento utilizada consta de los siguientes elementos:

- Un esquema de acción perceptual que compara a las celdas y a las respuestas de un problema, para producir una representación simbólica del mismo (Sección 2.1).
- Un esquema de acción perceptual que identifica igualdades entre las figuras de la matriz.

Una estrategia para resolver un problema de ILP consiste en iniciar generando relaciones completas pero inconsistentes, y luego refinarlas de modo que se vuelvan consistentes conservando su completitud. De este modo, las relaciones generadas y sus refinamientos producen un espacio de relaciones, y el problema de ILP se reduce a un problema de búsqueda en este espacio.

El modelo de inducción propuesto para resolver la RPM inicia generando reglas básicas, las cuales satisfacen a las figuras presentes en la matriz pero también pueden satisfacer a figuras ausentes. Luego, las reglas básicas se refinan para tratar de que satisfagan a menos figuras ausentes. Con esta estrategia, el problema de inducir reglas en la RPM se reduce a un problema de búsqueda en el espacio de reglas generado, lo cual es el tema de la siguiente sección.

2.3 Selección de reglas

El modelo de inducción presentado genera reglas con base en la información contenida en la matriz del problema, sin embargo, varias de las reglas generadas son redundantes, parcialmente incorrectas o irrelevantes para resolver el problema. Una vez que el modelo de inducción ha generado un espacio de reglas candidatas, es necesario realizar una búsqueda en este espacio para seleccionar al mejor conjunto de reglas.

Esta sección comenzará por proponer una definición del mejor conjunto de reglas, y luego presentará varios métodos para encontrar este conjunto. El uso de varios métodos de búsqueda permitió analizar las características del espacio de reglas, identificar la mejor forma de seleccionar reglas, y determinar en qué medida el desempeño del modelo depende de la definición del espacio de búsqueda, de la definición del mejor conjunto de reglas, y del método utilizado para encontrar este conjunto.

Desempeño de una regla

La Figura 2.13 muestra de nuevo a la regla de la Figura 2.10. La regla representada es $c_{k,1} = T(c_{k,0}) \cap T(c_{k+1,0})^C$, y las líneas discontinuas rojas muestran a la instancia $k = 0$ de esta regla. Esta instancia describe la hipótesis de que la celda superior derecha es igual a las figuras que están en la celda superior izquierda pero no están en la celda inferior izquierda, es decir, la hipótesis de que la celda superior derecha debe contener un triángulo pero no un círculo. Esta hipótesis es generada a partir del lado derecho de la instancia, calculando la intersección de los operandos positivos

($T(c_{k,0})$) y negativos ($T(c_{k+1,0})^C$). Por otro lado, la celda *resultado* de la instancia ($c_{0,1}$) se utilizará para validar esta hipótesis.

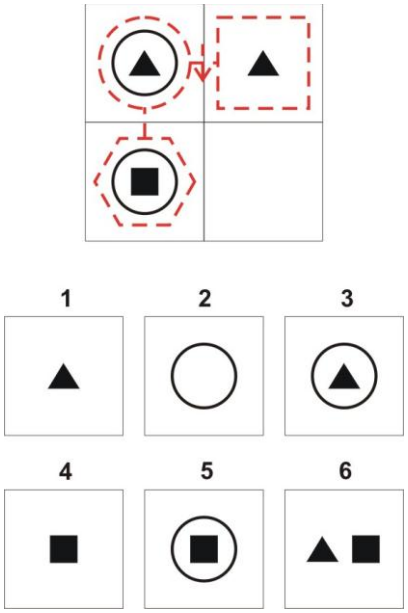


Figura 2.13. Regla compuesta que se extiende por filas.

De este modo, la hipótesis de una instancia consta de dos tipos de figuras:

- Figuras que se espera que estén en la celda resultado, llamadas *figuras positivas*. Estas figuras están en todos los operandos positivos de la instancia, pero en ninguno de los operandos negativos.
- Figuras que se espera que no estén en la celda resultado, llamadas *figuras negativas*. Estas figuras están en algún operando negativo de la instancia, y en al menos uno de los operandos positivos. O bien están en alguno de los operandos positivos, pero no en todos.

Las figuras que están en algún operando negativo, pero en ninguno de los operandos positivos, no influyen en el conjunto de figuras positivas y son descartadas de la hipótesis. Esta heurística busca ignorar figuras que son irrelevantes para una instancia en particular. Por ejemplo, en la instancia $k = 0$ de la Figura 2.13, el triángulo de la celda superior izquierda es una figura positiva, los círculos de la columna izquierda son figuras negativas, y el cuadrado es descartado de la hipótesis.

Una vez que se ha definido la hipótesis de una instancia, esta hipótesis se compara con la celda *resultado* de la instancia para generar cuatro medidas de desempeño:

- *Positivos verdaderos*: las figuras positivas que sí están en la celda resultado.
- *Falsos positivos*: las figuras positivas que no están en el resultado.
- *Negativos verdaderos*: las figuras negativas que no están en el resultado.
- *Falsos negativos*: las figuras negativas que sí están en el resultado.

Al sumar el número de positivos verdaderos y de negativos verdaderos se obtiene la *cobertura* de la instancia, mientras que al sumar el número de falsos positivos y de falsos negativos se obtienen los *errores* de la instancia. En la Figura 2.13, la instancia $k = 0$ cubre a los dos triángulos (positivos verdaderos) y a los dos círculos (negativos verdaderos), y no tiene errores. Nótese que cada figura de la matriz se cuenta por separado, aunque sea idéntica a otra figura.

Por otro lado, la instancia $k = 1$ de la Figura 2.13 involucra a la celda faltante, de modo que esta instancia se deja sin evaluar y el desempeño total de la regla es igual al desempeño de su instancia $k = 0$. Sin embargo, si la instancia $k = 1$ pudiera evaluarse generaría otras cuatro medidas de desempeño, las cuales se sumarían a las de la instancia $k = 0$ para obtener el desempeño total de la regla.

Sin embargo, ¿qué sucede si una misma figura aparece como negativo verdadero en la primera instancia, y como falso negativo en la segunda instancia? Por ejemplo, si la matriz de la Figura 2.13 fuera más grande y contuviera un círculo en lugar de la celda faltante, la instancia $k = 1$ de la regla podría evaluarse. En este caso, los círculos de la columna izquierda serían negativos verdaderos en la instancia $k = 0$, pero falsos negativos en la instancia $k = 1$. Para resolver esta clase de conflictos se utiliza la siguiente heurística de prioridades:

positivo verdadero > negativo verdadero > falso positivo > falso negativo

Es decir, si una figura era un positivo verdadero en la primera instancia, se deja como tal. Si era un negativo verdadero se deja como tal, a menos que sea un positivo verdadero en la segunda instancia, en cuyo caso la figura se convierte a un positivo verdadero. Y así sucesivamente. Si una regla tiene más de dos instancias, sus desempeños se combinan secuencialmente de la misma forma.

Por último, al combinar el desempeño de dos instancias se calculan otras cuatro medidas de desempeño:

- *Positivos verdaderos absolutos*: calculados al sumar directamente el número de positivos verdaderos en cada una de las instancias. Es decir, si una misma figura es un positivo verdadero en ambas instancias, se cuenta dos veces. En cambio, al contar los *positivos verdaderos* de la regla, cada figura se cuenta sólo una vez.
- *Negativos verdaderos absolutos*: la suma de los negativos verdaderos.
- *Falsos positivos absolutos*: la suma de los falsos positivos.
- *Falsos negativos absolutos*: la suma de los falsos negativos.

Complejidad de una regla

Además de preferir a las reglas con mayor cobertura y menos errores, se prefiere a las reglas más simples. La complejidad de una regla se define mediante tres características:

- Número de operandos, tanto positivos como negativos.
- Número de transformaciones diferentes de la identidad, es decir, el número de operandos que involucran traslación, escalamiento, rotación o reflejo.
- Irregularidad, definida como el número de columnas que contienen a la celda resultado (en reglas que se extienden por filas), o el número de filas que contienen a la celda resultado (en reglas que se extienden por columnas).

La celda resultado de una regla que se extiende por filas siempre está en la misma columna, y la celda resultado de una regla que se extiende por columnas siempre está en la misma fila, de modo que la irregularidad de cualquier regla es 1. Sin embargo, la irregularidad será una característica útil al abordar la complejidad de un conjunto de reglas.

Costo de una regla

A partir de las medidas de desempeño y de complejidad de una regla es posible definir una función de costo. En este proyecto, el costo de una regla *r* se definió empíricamente como:

$$\begin{aligned} costo(r) = {} & traslapePositivo + 15*sobrantes + 5*sobrantesPositivos \\ & + 3*irregularidad + 2*operandos + 2*transformaciones \\ & + 2*falsosPositivos + falsosPositivosAbsolutos + falsosNegativosAbsolutos \\ & + sobrantesPositivos*traslapePositivo \end{aligned} \tag{2.1}$$

Donde:

traslapePositivo = *positivosVerdaderosAbsolutos* − *positivosVerdaderos*
sobrantes = *totalDeFiguras* − *cobertura*
sobrantesPositivos = *totalDeFiguras* − *positivosVerdaderos*
totalDeFiguras es el número de figuras en la matriz (sean iguales o diferentes)
cobertura = *positivosVerdaderos* + *negativosVerdaderos*
y *transformaciones* es el número de transformaciones diferentes de la identidad

Esta función expresa la idea de que las mejores reglas producen la mayor cobertura, evitando cubrir varias veces a la misma figura, produciendo el menor número de errores y requiriendo la menor complejidad.

Además, los positivos verdaderos se prefieren sobre los negativos verdaderos, ya que se busca que las reglas expliquen por qué una figura aparece en una celda, y no tanto por qué una figura no aparece en una celda.

El producto *sobrantesPositivos*traslapePositivo* indica que el traslape positivo recibe una mayor penalización cuando la regla presenta además sobrantes positivos, lo cual sugiere que la cobertura de la regla a lo largo de la matriz no es uniforme.

Conjuntos de reglas

Algunos problemas de la RPM pueden resolverse con una sola regla, sin embargo, en general es necesario un conjunto de reglas para poder describir la lógica de la matriz. Es por esto que las ocho medidas de desempeño, las tres medidas de complejidad y la función de costo definidas para reglas individuales también se definen para conjuntos de reglas.

Las medidas de desempeño de un conjunto de reglas se calculan combinando las medidas de las reglas individuales, del mismo modo en el que se combinó el desempeño de varias instancias para obtener el desempeño total de una regla. Por ejemplo, los *positivos verdaderos* del conjunto se obtienen combinando los *positivos verdaderos* de las reglas, siguiendo la heurística de prioridades. Del mismo modo, los *positivos verdaderos absolutos* del conjunto se calculan sumando directamente el número de *positivos verdaderos* de las reglas (no sumando los *positivos verdaderos absolutos* de las reglas).

Las medidas de complejidad de un conjunto de reglas se calculan sumando directamente las medidas de las reglas individuales. En este caso, la medida de irregularidad puede ser un valor entre 1 y $m + n$, donde m y n son el número de filas y el número de columnas en la matriz del problema, respectivamente.

El costo de un conjunto de reglas también se calcula con la Ecuación 2.1, utilizando las medidas de desempeño y complejidad del conjunto.

Selección individual

Una estrategia para encontrar al mejor conjunto de reglas es seleccionar individualmente a las mejores reglas hasta que el conjunto formado alcance un desempeño satisfactorio. En este proyecto se implementaron tres algoritmos con esta estrategia: uno codicioso, uno avaro y uno aleatorio.

Algoritmo codicioso

En cada iteración, este algoritmo selecciona a la regla que produzca el máximo incremento en el número de positivos verdaderos del conjunto seleccionado, y que tenga el menor costo individual.

Entrada: Un conjunto de reglas candidatas C, un límite de reglas n, y un total de figuras f.
Salida: Un subconjunto S de C.

1. Hacer $S \leftarrow \emptyset$.
2. Si C está vacío, devolver S y terminar.
3. Buscar en C a la regla r que, al agregarse a S, produzca el máximo incremento en el número de positivos verdaderos de S. En caso de empate, seleccionar a la regla con el menor costo individual. Si el empate persiste, seleccionar a la primera regla encontrada.
4. Si el incremento producido por r es cero, devolver S y terminar.
5. Agregar r a S.
6. Si el número de reglas en S es igual a n, o si el número de positivos verdaderos en S es igual a f, devolver S y terminar.
7. Regresar al paso 3.

Algoritmo avaro

En cada iteración, este algoritmo selecciona a la regla con el menor costo individual, siempre que ésta produzca algún incremento en el número de positivos verdaderos del conjunto seleccionado. Es similar al algoritmo codicioso, pero le da más prioridad al costo individual que al número de positivos verdaderos del conjunto.

Entrada: Un conjunto de reglas candidatas C, un límite de reglas n, y un total de figuras f.
Salida: Un subconjunto S de C.

1. Hacer $S \leftarrow \emptyset$.
2. Si C está vacío, devolver S y terminar.
3. Si r es la regla con el menor costo individual en C, y al agregar r a S se produce algún incremento en el número de positivos verdaderos de S, entonces agregar r a S.
4. Eliminar a r de C.

5. Si el número de reglas en S es igual a n, o si el número de positivos verdaderos en S es igual a f, devolver S y terminar.
6. Regresar al paso 2.

Algoritmo aleatorio

Este algoritmo simplemente forma un conjunto aleatorio de reglas, y sólo se utilizó para establecer un desempeño mínimo que los demás algoritmos deberían sobrepasar.

Entrada: Un conjunto de reglas candidatas C, y un límite de reglas n.
Salida: Un subconjunto S de C.

1. Definir k como un número aleatorio en el intervalo [1,n].
2. Definir S seleccionando aleatoriamente k reglas diferentes de C.
3. Devolver S y terminar.

Selección de conjunto

El espacio de reglas candidatas y la función de costo definida para conjuntos de reglas permiten buscar al mejor conjunto utilizando algoritmos de búsqueda local. En este proyecto se implementaron tres de estos algoritmos: pasos descendentes, búsqueda tabú y recocido simulado.

Estos métodos parten de una solución inicial aleatoria, y en cada iteración exploran la *vecindad* de la solución actual para tratar de encontrar una solución mejor. En el problema en cuestión, una solución S se define como un subconjunto del conjunto de reglas candidatas C. Además, la vecindad de una solución S se define como todas las soluciones que pueden obtenerse a partir de S realizando uno de los siguientes *movimientos*:

- Agregar una sola regla r a S (tal que $r \in C$ y $r \notin S$).
- Quitar una sola regla r a S.

En los siguientes algoritmos se descartan las soluciones vacías o que contienen más de n reglas, donde n se define experimentalmente.

Pasos descendentes

Este algoritmo pasa al mejor vecino de la solución actual, si este vecino es mejor que la solución actual.

Entrada: Un conjunto de reglas candidatas C.
Salida: Un subconjunto S de C.

1. Definir S como un subconjunto aleatorio de C.
2. Encontrar a la solución V con el menor costo en la vecindad de S.
3. Si costo(V) < costo(S), hacer $S \leftarrow V$ y regresar al paso 2.
4. Devolver S y terminar.

Búsqueda tabú

Este algoritmo pasa al mejor vecino de la solución actual, si el *movimiento* que permite pasar a este vecino no está en una lista de movimientos restringidos (*lista tabú*), o si este vecino es mejor que la mejor solución encontrada hasta el momento. En este algoritmo la lista tabú se implementa como una lista circular.

Entrada: Un conjunto de reglas candidatas C, un límite de iteraciones m, un costo satisfactorio c, y un tamaño límite t para la lista tabú.
Salida: Un subconjunto S de C.

1. Definir S como un subconjunto aleatorio de C.
2. Hacer $M \leftarrow S$ para definir a la mejor solución encontrada hasta el momento.
3. Crear una lista vacía L de tamaño t, y hacer $i \leftarrow 0$.
4. Buscar a la solución V con el menor costo en la vecindad de S, tal que V cumpla alguna de las siguientes condiciones:
   a) El movimiento para obtener V a partir de S no está en la lista L.
   b) costo(V) < costo(M).
5. Si no se encuentra tal solución V, devolver M y terminar.
6. Si costo(V) < costo(M), hacer $M \leftarrow V$.
7. Si la lista L está llena, eliminar a su elemento más antiguo.
8. Agregar a L el movimiento que permite obtener V a partir de S.
9. Hacer $S \leftarrow V$, y $i \leftarrow i + 1$.
10. Si costo(M) < c, o si $i = m$, devolver M y terminar.
11. Regresar al paso 4.

Recocido simulado

Este algoritmo está inspirado en el proceso metalúrgico del recocido, que consiste en calentar una pieza de metal y enfriarla lentamente. El algoritmo utiliza una variable de *temperatura* que es alta al inicio y se reduce lentamente. Conforme la temperatura decrece, la probabilidad de que el algoritmo pase a un estado con un costo mayor que el del estado actual también decrece. Por lo tanto, el método se comporta al inicio de modo similar a una búsqueda aleatoria, y al final de modo similar a una búsqueda por pasos descendentes.

Entrada: Un conjunto de reglas candidatas C, un límite de iteraciones m, un costo satisfactorio c, una temperatura inicial T_0, un factor de enfriamiento $u \in (0,1)$, y un intervalo de enfriamiento k.
Salida: Un subconjunto S de C.

1. Definir S como un subconjunto aleatorio de C.
2. Hacer $M \leftarrow S$ para definir a la mejor solución encontrada hasta el momento.
3. Hacer $T \leftarrow T_0$, $i \leftarrow 0$, y $j \leftarrow 0$.
4. Definir a V como un vecino aleatorio de S.
5. Si costo(V) < costo(M), hacer $M \leftarrow V$.
6. Hacer $\Delta \leftarrow$costo(V) – costo(S).
7. Si $\Delta \leq 0$, hacer $S \leftarrow V$ e ir al paso 10.
8. Definir p como un valor aleatorio en el intervalo (0,1).
9. Si $p < e^{-\Delta/T}$, hacer $S \leftarrow V$ (en este paso, e es la constante matemática 2.71828...).
10. Hacer $i \leftarrow i + 1$, y $j \leftarrow j + 1$.

11. Si $j = k$, hacer $T \leftarrow uT$, y $j \leftarrow 0$.
12. Si costo(M) < c, o si $i = m$, devolver M y terminar.
13. Regresar al paso 4.

Selección híbrida

Finalmente, se implementaron dos algoritmos que combinan un algoritmo de selección individual con uno de selección de conjunto:

- Realizar el algoritmo codicioso, y utilizar la solución resultante como la solución inicial del algoritmo de pasos descendentes.
- Realizar el algoritmo avaro, y utilizar la solución resultante como la solución inicial del algoritmo de pasos descendentes.

La aplicación posterior del algoritmo de pasos descendentes permite reducir el costo de la solución generada, usualmente eliminando reglas redundantes. Además, en cada caso se probaron dos variantes del algoritmo de pasos descendentes:

a) Permitiendo que el algoritmo agregara o eliminara reglas del conjunto seleccionado.
b) Permitiendo que el algoritmo sólo eliminara reglas del conjunto seleccionado.

2.4 Selección de la respuesta

En el análisis de la Sección 1.2 se observó que pueden utilizarse dos estrategias diferentes para seleccionar la respuesta de un problema:

a) Generar la celda faltante de la matriz (total o parcialmente) y luego compararla con las posibles respuestas.
b) Insertar cada posible respuesta en la matriz para asignarle un puntaje.

En este proyecto se implementaron ambas estrategias. En la primera, la matriz es analizada sin considerar a las posibles respuestas, induciendo y seleccionando reglas con los métodos presentados en las secciones 2.2 y 2.3. Las reglas seleccionadas representan hipótesis sobre el contenido de las celdas faltantes de la matriz, de modo que estas reglas generan total o parcialmente, y de forma implícita, a las celdas faltantes. Luego, las posibles respuestas se asignan a la o las celdas faltantes de la matriz, y se elige la solución que produzca el mayor incremento en la cobertura de las reglas seleccionadas; en caso de empate, se elige la solución que produzca la mayor cobertura absoluta (*positivos verdaderos absolutos* mas *negativos verdaderos absolutos*); si el empate persiste, se elige la solución con el menor número de figuras. Esta estrategia se basa en la idea de que la mejor solución es la que satisface el mayor número de reglas con el menor número de figuras.

En la segunda estrategia, las posibles respuestas se asignan a la o las celdas faltantes de la matriz, generando un conjunto de matrices candidatas sin celdas faltantes. Para cada una de estas matrices completas se induce y se selecciona un conjunto de reglas, usando los métodos de las secciones 2.2 y 2.3. Luego, el costo de cada matriz completa se define como el costo del conjunto de reglas seleccionado para esta matriz. En otras palabras, el costo de una matriz se define como el costo de las reglas necesarias para describirla. Finalmente, se elige la solución que produjo la matriz

completa con el menor costo. Esta estrategia se basa en la idea de que la mejor solución es la que permite describir a la matriz de la forma más simple posible.

2.5 Resultados en la prueba estándar

Como se implementaron ocho algoritmos de selección de reglas y dos algoritmos de selección de la respuesta, el modelo tiene 16 variantes básicas. A continuación se reporta el desempeño de cada variante al resolver los 60 problemas de la RPM estándar, los cuales se codificaron manualmente utilizando el lenguaje descrito en la Sección 2.1. Luego, al final de esta sección, se presenta un análisis de los resultados obtenidos.

Debido a la confidencialidad de la RPM no se reportan las reglas inducidas en cada problema, sino sólo el conteo de aciertos. Sin embargo, la Sección 2.6 abordará problemas fabricados y sí describirá las reglas inducidas.

El programa fue implementado en el lenguaje Java SE 1.6.0_20. Los tiempos de ejecución reportados se obtuvieron en una laptop con 1.12 GB de RAM y procesador AMD Sempron a 1.58 GHz. Estos tiempos indican cuánto le llevó al programa resolver los 60 problemas de la prueba; como se mencionó anteriormente, la prueba tiene 5 secciones etiquetadas de la A a la E, y 12 problemas en cada sección.

Generando la celda faltante

Algoritmo aleatorio

Este algoritmo se probó con reglas de nivel 2, y luego con reglas de nivel 2 y 3. El tamaño del conjunto seleccionado se fijó en 1, 10 y 40 reglas, de modo que se probaron 6 variantes de este algoritmo, cada una de las cuales se ejecutó 10 veces. La Tabla 2.1 muestra el tiempo de ejecución promedio de cada variante y el promedio de problemas correctos en cada sección de la prueba.

Tabla 2.1. Resultados del algoritmo aleatorio (generando la celda faltante).

Nivel	Tamaño	Tiempo (s)	A	B	C	D	E	Total
2	1	1	6.8	4.8	1.3	1.2	1.5	15.6
2	10	1	10.9	6.9	5.1	6	2.2	31.1
2	40	1	11	7	6.5	7	2.9	34.4
2 y 3	1	1	3.5	4.9	1.4	1.5	1.4	12.7
2 y 3	10	2	6	7.9	3.9	7	1.9	26.7
2 y 3	40	15	6	9	6.4	7.2	3.4	32

A pesar de seleccionar reglas aleatoriamente, este algoritmo alcanzó un promedio de 34.4 problemas correctos, lo cual sugiere que en muchos problemas la mayoría de las reglas inducidas son correctas y la probabilidad de elegir reglas erróneas es baja.

El desempeño del algoritmo mejora conforme se incrementa el tamaño del conjunto seleccionado. Sin embargo, para un mismo tamaño del conjunto, la variante de nivel 2 y 3 obtuvo un resultado total más bajo que la variante de nivel 2, principalmente debido a errores en la sección A.

Algoritmo codicioso

La Tabla 2.2 muestra los resultados de este algoritmo.

Tabla 2.2. Resultados del algoritmo codicioso (generando la celda faltante).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	1	11	6	8	8	5	38
2 y 3	3	4	8	6	9	7	34

La variante de nivel 2 obtuvo mejores resultados que el algoritmo aleatorio, pero la variante de nivel 2 y 3 fue ligeramente inferior al algoritmo aleatorio, debido a errores en la sección A.

Algoritmo avaro

La Tabla 2.3 muestra los resultados de este algoritmo.

Tabla 2.3. Resultados del algoritmo avaro (generando la celda faltante).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	1	11	6	7	8	5	37
2 y 3	4	4	8	6	9	7	34

La variante de nivel 2 fue superior al algoritmo aleatorio, mientras que la de nivel 2 y 3 fue ligeramente inferior al algoritmo aleatorio, debido a errores en la sección A.

Pasos descendentes

Cada variante de este algoritmo se ejecutó 10 veces. El promedio de sus resultados se muestra en la Tabla 2.4.

Tabla 2.4. Resultados del algoritmo de pasos descendentes (generando la celda faltante).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	2	10.6	6.5	7.4	7.6	3.8	35.9
2 y 3	5	10.5	7.7	5.9	8.1	2.3	34.5

Los resultados apenas superan a los del algoritmo aleatorio, lo cual sugiere que el espacio de búsqueda contiene muchos mínimos locales. En este caso ambas variantes obtuvieron un desempeño similar, y sus resultados en la sección A fueron casi idénticos.

Búsqueda tabú

Los parámetros de este algoritmo se ajustaron experimentalmente: 20 para el tamaño de la lista tabú, 100 para el límite de iteraciones y 0 para el costo satisfactorio. Cada variante de este algoritmo se ejecutó 10 veces. El promedio de sus resultados se muestra en la Tabla 2.5.

Tabla 2.5. Resultados de búsqueda tabú (generando la celda faltante).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	8	10.7	6.8	8.1	7.5	4.2	37.3
2 y 3	30.1	9.5	7.9	7.7	9	2.9	37

Los resultados superan a los del algoritmo aleatorio. Ambas variantes obtuvieron un resultado total similar, pero la variante de nivel 2 y 3 requirió un mayor tiempo de ejecución. En la sección A, la variante de nivel 2 sólo fue ligeramente superior a la de nivel 2 y 3.

Recocido simulado

Los parámetros de este algoritmo se ajustaron experimentalmente. La temperatura inicial se definió como $T_0 =$ *costoMáximo* / *ln(0.99)*, donde:

*costoMáximo = traslapePositivo + 15*sobrantes + 5*sobrantesPositivos*
*+ 3*irregularidad + 2*operandos + 2*transformaciones*
*+ 2*falsosPositivos + falsosPositivosAbsolutos + falsosNegativosAbsolutos*
*+ sobrantesPositivos*traslapePositivo*
traslapePositivo = 0
sobrantes = sobrantesPositivos = totalDeFiguras
irregularidad = límiteDeReglas
*operandos = transformaciones = 2*límiteDeReglas*max(numFilas, numColumnas)*
falsosPositivos = falsosPositivosAbsolutos = totalDeFiguras
falsosNegativosAbsolutos = 0
totalDeFiguras es el número de figuras en la matriz (iguales o diferentes)
límiteDeReglas es el tamaño máximo del conjunto de reglas seleccionado, fijado en 20
numFilas y *numColumnas* son el número de filas y columnas en la matriz, respectivamente

La variable *costoMáximo* es una estimación del costo máximo que puede alcanzar cualquier conjunto de reglas en un problema dado, y se definió con base en la función de costo (Ecuación 2.1). Con esta temperatura inicial, el algoritmo es capaz de pasar a soluciones con costo *costoMáximo* con una probabilidad de 0.99. Luego, conforme la variable *temperatura* decrece, el algoritmo se vuelve más selectivo.

El costo satisfactorio se fijó en 0, el límite de iteraciones en 100,000 y el intervalo de enfriamiento en 10,000. Por último, el factor de enfriamiento se definió como $u = \left(\frac{T_f}{T_0}\right)^{1/99}$, donde $T_f = 10^{-10}$. De este modo, la variable *temperatura* toma 100 valores diferentes, iniciando en T_0 y terminando en $T_f = T_0 u^{99} = 10^{-10}$.

Cada variante de este algoritmo se ejecutó 10 veces. El promedio de sus resultados se muestra en la Tabla 2.6.

Tabla 2.6. Resultados de recocido simulado (generando la celda faltante).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	26	10.8	6.4	8	7.6	4.9	37.7
2 y 3	51.5	3.3	7	6.7	9.4	2.9	29.3

La variante de nivel 2 superó al algoritmo aleatorio, pero la de nivel 2 y 3 fue considerablemente inferior, principalmente debido a errores en la sección A.

Algoritmo codicioso con pasos descendentes

Cada variante de este algoritmo se probó con dos versiones del algoritmo de pasos descendentes:

a) Permitiendo que el algoritmo agregara o eliminara reglas del conjunto seleccionado.
b) Permitiendo que el algoritmo sólo eliminara reglas del conjunto seleccionado.

La Tabla 2.7 muestra los resultados de cada variante.

Tabla 2.7. Resultados del algoritmo codicioso con pasos descendentes (generando la celda faltante).

Nivel	Pasos descendentes	Tiempo (s)	A	B	C	D	E	Total
2	a	1	11	6	8	8	5	38
2	b	1	11	6	8	8	5	38
2 y 3	a	4	4	8	6	9	7	34
2 y 3	b	4	4	8	6	9	7	34

El desempeño fue el mismo para las versiones *a* y *b* del algoritmo de pasos descendentes, tanto en la variante de nivel 2 como en la de nivel 2 y 3, lo cual sugiere que pasos descendentes tiende a eliminar reglas a partir de la solución generada por el algoritmo codicioso, y que no suele agregar reglas a esta solución.

La variante de nivel 2 superó al algoritmo aleatorio, mientras que la de nivel 2 y 3 fue ligeramente inferior al algoritmo aleatorio, debido a errores en la sección A.

Algoritmo avaro con pasos descendentes

Como en el algoritmo anterior, cada variante de este algoritmo se probó con las versiones *a* y *b* del algoritmo de pasos descendentes. Los resultados se muestran en la Tabla 2.8.

Tabla 2.8. Resultados del algoritmo avaro con pasos descendentes (generando la celda faltante).

Nivel	Pasos descendentes	Tiempo (s)	A	B	C	D	E	Total
2	a	1	11	6	7	8	5	37
2	b	1	11	6	7	8	5	37
2 y 3	a	4	4	8	6	10	7	35
2 y 3	b	4	4	8	6	10	6	34

Las versiones *a* y *b* del algoritmo de pasos descendentes produjeron resultados idénticos en la variante de nivel 2, pero diferentes en la variante de nivel 2 y 3. Nuevamente, la variante de nivel 2 fue superior a la de nivel 2 y 3, debido a errores en la sección A.

Insertando las posibles respuestas

Algoritmo aleatorio

Como en el método *generando la celda faltante*, este algoritmo se probó con reglas de nivel 2 y con reglas de nivel 2 y 3, fijando el tamaño del conjunto seleccionado a 1, 10 y 40, y ejecutando cada variante 10 veces. La Tabla 2.9 muestra el promedio de los resultados obtenidos.

Tabla 2.9. Resultados del algoritmo aleatorio (insertando las posibles respuestas).

Nivel	Tamaño	Tiempo (s)	A	B	C	D	E	Total
2	1	8	3.7	2.5	1	1.9	2.1	11.2
2	10	8	6.9	5.8	2.1	4.5	2.1	21.4
2	40	9	4	5	1.8	4.4	2.4	17.6
2 y 3	1	18	3.3	3.1	1.2	1.7	2.1	11.4
2 y 3	10	40	5.9	6.1	1.6	4.5	1.7	19.8
2 y 3	40	216.5	3	9.1	2.3	5	0.9	20.3

El desempeño fue más bajo que con el método *generando la celda faltante*., lo cual indica que la selección de reglas juega un papel más importante en el método *insertando las posibles respuestas*. En este caso, las variantes de nivel 2 y de nivel 2 y 3 obtuvieron un desempeño muy similar, y el desempeño no varió de forma consistente al cambiar el tamaño del conjunto seleccionado. El mejor resultado promedio de este algoritmo fue 21.4.

Algoritmo codicioso

Los resultados de este algoritmo se muestran en la Tabla 2.10.

Tabla 2.10. Resultados del algoritmo codicioso (insertando las posibles respuestas).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	9	11	8	4	3	7	33
2 y 3	46	11	12	8	11	10	52

Ambas variantes superaron al algoritmo aleatorio, y la variante de nivel 2 y 3 obtuvo un mejor desempeño.

Algoritmo avaro

La Tabla 2.11 muestra los resultados de este algoritmo.

Tabla 2.11. Resultados del algoritmo avaro (insertando las posibles respuestas).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	9	11	9	5	5	6	36
2 y 3	50	11	10	8	9	8	46

Ambas variantes superaron al algoritmo aleatorio, y la variante de nivel 2 y 3 obtuvo un mejor desempeño.

Pasos descendentes

Cada variante de este algoritmo se ejecutó 10 veces. El promedio de sus resultados se muestra en la Tabla 2.12.

Tabla 2.12. Resultados del algoritmo de pasos descendentes (insertando las posibles respuestas).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	28	9.3	8	2.3	2.1	4.9	26.6
2 y 3	90.1	7	8.3	3.4	4.1	4	26.8

Ambas variantes obtuvieron un desempeño similar y superaron ligeramente al algoritmo aleatorio.

Búsqueda tabú

Los parámetros de este algoritmo se ajustaron experimentalmente: 20 para el tamaño de la lista tabú, 100 para el límite de iteraciones y 0 para el costo satisfactorio. Cada variante se ejecutó 10 veces. La Tabla 2.13 muestra el promedio de sus resultados.

Tabla 2.13. Resultados de búsqueda tabú (insertando las posibles respuestas).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	168	10.4	9	2.7	2.2	6.2	30.5
2 y 3	788.6	9.7	10.5	4.4	5.6	7.2	37.4

Ambas variantes superaron al algoritmo aleatorio, y la variante de nivel 2 y 3 obtuvo un mejor desempeño.

Recocido simulado

Los parámetros de este algoritmo se ajustaron experimentalmente, del mismo modo que en el método *generando la celda faltante*, pero en este caso el límite de iteraciones se fijó en 10,000 y el intervalo de enfriamiento en 1,000. Cada variante se ejecutó 10 veces. El promedio de sus resultados se muestra en la Tabla 2.14.

Tabla 2.14. Resultados de recocido simulado (insertando las posibles respuestas).

Nivel	Tiempo (s)	A	B	C	D	E	Total
2	31	10.7	7.7	2.3	1.9	4.6	27.2
2 y 3	90.4	11	9.9	4	8.4	4.4	37.7

Ambas variantes superaron al algoritmo aleatorio, y la variante de nivel 2 y 3 obtuvo un mejor desempeño.

Algoritmo codicioso con pasos descendentes

Como en el método *generando la celda faltante*, cada variante de este algoritmo se probó con las versiones *a* y *b* del algoritmo de pasos descendentes. Los resultados se muestran en la Tabla 2.15.

Tabla 2.15. Resultados del algoritmo codicioso con pasos descendentes (insertando las posibles respuestas).

Nivel	Pasos descendentes	Tiempo (s)	A	B	C	D	E	Total
2	a	11	11	8	4	3	7	33
2	b	8	11	8	4	3	7	33
2 y 3	a	43	11	12	10	10	10	53
2 y 3	b	32	11	12	10	11	10	54

Las cuatro variantes superaron al algoritmo aleatorio, y la variante de nivel 2 y 3 obtuvo un mejor desempeño al combinarse con la versión *b* del algoritmo de pasos descendentes.

Algoritmo avaro con pasos descendentes

Cada variante de este algoritmo se probó con las versiones *a* y *b* del algoritmo de pasos descendentes. Los resultados se muestran en la Tabla 2.16.

Tabla 2.16. Resultados del algoritmo avaro con pasos descendentes (insertando las posibles respuestas).

Nivel	Pasos descendentes	Tiempo (s)	A	B	C	D	E	Total
2	a	13	11	9	3	3	6	32
2	b	9	11	9	3	3	6	32
2 y 3	a	50	11	10	7	9	7	44
2 y 3	b	29	11	10	7	9	6	43

Las cuatro variantes superaron al algoritmo aleatorio, y la variante de nivel 2 y 3 obtuvo mejores resultados al combinarse con la versión *a* del algoritmo de pasos descendentes.

Análisis

Generando la celda faltante

La Tabla 2.17 muestra el mejor desempeño de cada algoritmo al utilizar el método *generando la celda faltante*. El algoritmo aleatorio con tamaño 40 estableció un desempeño base relativamente alto que no fue superado de forma considerable por los demás algoritmos. Sin embargo, los demás algoritmos seleccionaron menos de 40 reglas para resolver cada problema. Por ejemplo, la Tabla 2.18 muestra el promedio de reglas seleccionadas por el algoritmo codicioso.

Tabla 2.17. Variantes con mejor desempeño (generando la celda faltante).

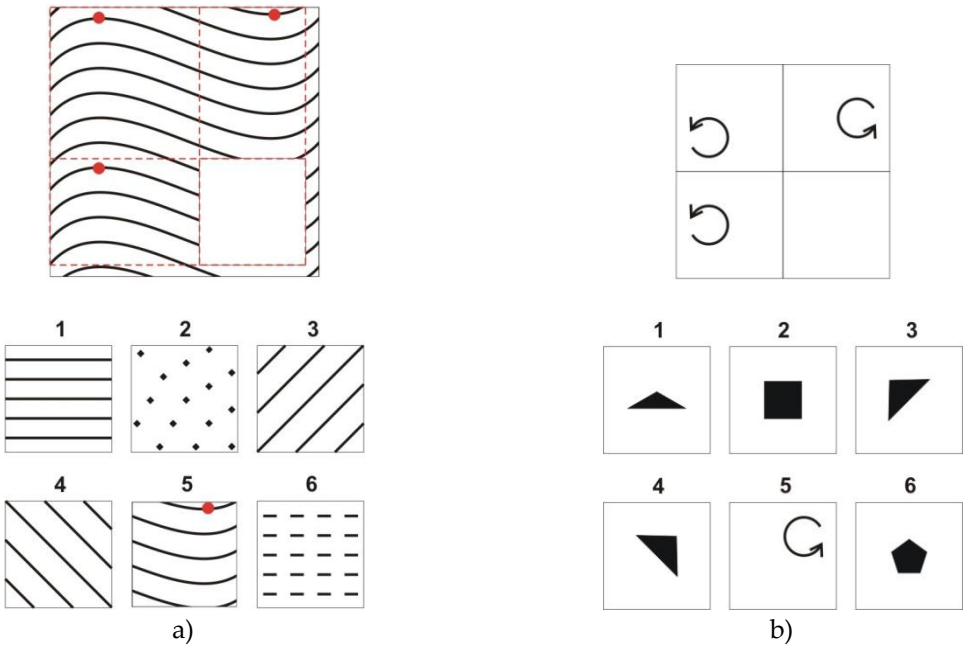
Selección de reglas	Nivel	Total
Aleatorio con tamaño 40	2	34.4
Codicioso	2	38
Avaro	2	37
Pasos descendentes	2	35.9
Búsqueda tabú	2	37.3
Recocido simulado	2	37.7
Codicioso con pasos descendentes *a* o *b*	2	38
Avaro con pasos descendentes *a* o *b*	2	37

Tabla 2.18. Promedio de reglas seleccionadas por el algoritmo codicioso (generando la celda faltante).

Nivel	A	B	C	D	E	Promedio global
2	2	2.3	6.3	4.7	4.3	3.9
2 y 3	1.3	1.6	4.7	3.3	4.1	3

Es decir, el algoritmo seleccionó en promedio 3 o 4 reglas para resolver cada problema. La variante de nivel 2 y 3 utilizó menos reglas que la de nivel 2. Además, el promedio de reglas fue más bajo en las secciones A y B, las cuales sólo contienen matrices de 2x2 y son las más sencillas de la prueba.

Con el método *generando la celda faltante*, la variante de nivel 2 superó a la de nivel 2 y 3 en todos los algoritmos, generalmente debido a que esta última cometió más errores en la sección A. Esta sección es la más sencilla de la prueba y sólo contiene problemas con una sola imagen, sin embargo, el método utilizado para representar estos problemas facilita la inducción de reglas erróneas de nivel 3. Por ejemplo, al representar el problema de la Figura 2.14a con el método descrito en la Sección 2.1, el problema se convierte en el de la Figura 2.14b.



a) b)

Figura 2.14. Representación de un problema con una sola imagen. a) Problema original. b) Problema equivalente.

Al analizar la matriz del problema resultante, el programa puede inducir la regla de nivel 3 $c_{k,0} = T_1(c_{k,1}) \cap T_2(c_{k+1,0})$, donde T_1 es la transformación "rotar 180° y desplazar hacia la izquierda y hacia abajo" y T_2 es la transformación "desplazar hacia abajo". Aunque la instancia $k = 0$ de esta regla permite describir a todas las figuras de la matriz sin errores, la instancia $k = 1$ no puede satisfacerse con ninguna de las respuestas disponibles. El método *generando la celda faltante* selecciona reglas sin considerar a las posibles respuestas y no es capaz de cambiar esta selección, de modo que el programa no logra resolver este problema correctamente.

Esta deficiencia afectó el desempeño de todos los algoritmos, pues el desempeño máximo obtenido fue de 38. Este problema podría solucionarse permitiendo que el método reconsiderara su selección de reglas tras analizar a las posibles respuestas, o bien utilizando el método *insertando las posibles respuestas*, cuyos resultados se analizan a continuación.

Insertando las posibles respuestas

La Tabla 2.19 muestra el mejor desempeño de cada algoritmo al utilizar el método *insertando las posibles respuestas*. El desempeño del algoritmo aleatorio fue más bajo que en el método anterior, debido a que en este caso la selección de reglas juega un papel más importante. Mientras que en el método anterior se elige a la respuesta que satisface mejor al conjunto de reglas seleccionado, en este método se elige a la respuesta que produce el conjunto de reglas más simple.

Tabla 2.19. Variantes con mejor desempeño (insertando las posibles respuestas).

Selección de reglas	Nivel	Total
Aleatorio con tamaño 10	2	21.4
Codicioso	2 y 3	52
Avaro	2 y 3	46
Pasos descendentes	2 y 3	26.8
Búsqueda tabú	2 y 3	37.4
Recocido simulado	2 y 3	37.7
Codicioso con pasos descendentes *b*	2 y 3	54
Avaro con pasos descendentes *a*	2 y 3	44

Como sucedió con el método anterior, las dos variantes del algoritmo codicioso obtuvieron el mejor desempeño. Sin embargo, en este caso las variantes de nivel 2 y 3 superaron a las de nivel 2 en todos los algoritmos, exceptuando al aleatorio.

El mayor desempeño fue de 54, lo cual demuestra que, bajo el modelo propuesto, prácticamente toda la RPM estándar puede resolverse mediante reglas de nivel 2 o 3. Por otro lado, el programa no logró resolver seis problemas, los cuales sugieren mejoras que pueden agregarse al modelo en trabajos futuros:

- Uno en la sección C y uno en la sección E, que requieren refinar el proceso de selección de reglas.
- Uno en la sección A que requiere identificar la intersección de figuras (Figura 2.15).
- Uno en la sección C que requiere identificar transformaciones cualitativas (Figura 2.16).
- Uno en la sección D que requiere identificar deformaciones de figuras (Figura 2.17).
- Uno en la sección E que requiere interpretar figuras como números (Figura 2.18).

Figura 2.15. Intersección de figuras.

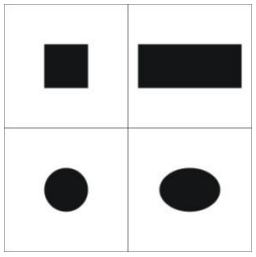


Figura 2.16. Transformaciones cualitativas. Aunque el cuadrado y el círculo se alargan, no lo hacen en la misma proporción.

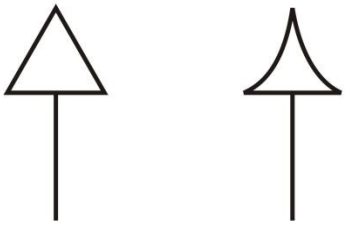


Figura 2.17. Deformación de figuras.

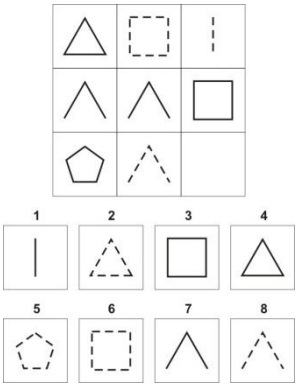


Figura 2.18. Interpretación de figuras como números. La respuesta es 4.

2.6 Resultados en problemas más generales

El modelo propuesto es capaz de resolver problemas más generales que los de la RPM, por ejemplo:

- Problemas con matrices de tamaño arbitrario, ya sean cuadradas o rectangulares.
- Problemas cuya matriz contiene varias celdas faltantes.
- Problemas que requieren inducir reglas que relacionen a más de tres celdas (es decir, reglas con nivel mayor a 3).
- Problemas que requieren inducir reglas que relacionen a celdas arbitrarias de la matriz, sin importar si estas celdas están en la misma fila o columna.
- Problemas que requieren inducir cualquier regla que pueda expresarse mediante los operadores de unión, intersección, complemento y transformación.

El problema de la Figura 2.19 ejemplifica varias de las características anteriores, y su representación en el lenguaje propuesto puede consultarse en el Apéndice B. A continuación se describen los resultados del programa al resolver este problema, induciendo reglas de nivel 2, 3 y 4, y utilizando el algoritmo codicioso con pasos descendentes *b* (*insertando las posibles respuestas*), que fue la variante con mejor desempeño. Al final de esta sección se describe la regla con base en la cual se diseñó este problema, y que se esperaba fuera inducida por el programa.

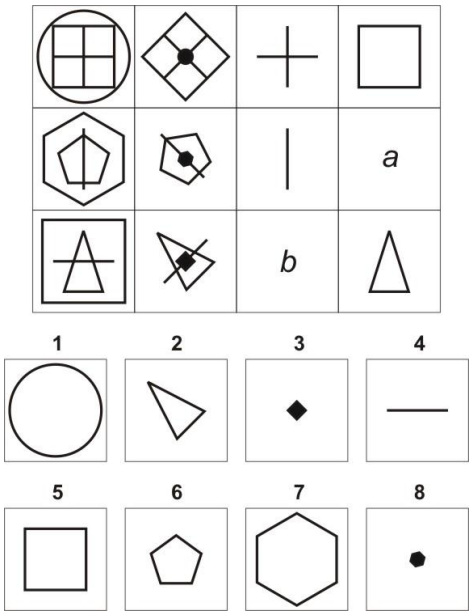


Figura 2.19. Problema más general que los de la RPM. La solución es [6,4] (*a* = 6 y *b* = 4).

Reglas de nivel 2

Al permitir sólo reglas de nivel 2, el programa produjo la matriz de costo de la Tabla 2.20. En esta matriz, la celda en la fila *m* y columna *n* contiene el costo de la solución [*m*,*n*], es decir, el costo del conjunto de reglas seleccionado al insertar la respuesta *m* en la celda *a* y la respuesta *n* en la celda *b*.

Tabla 2.20. Matriz de costo para reglas de nivel 2.

	1	2	3	4	5	6	7	8
1	163	185	185	**160**	179	184	184	174
2	182	179	206	181	200	202	202	195
3	182	206	179	181	181	202	199	195
4	182	206	206	181	184	205	181	202
5	201	227	227	195	219	209	209	231
6	171	195	195	170	189	194	194	184
7	171	195	195	170	189	194	194	186
8	198	218	218	195	216	218	218	179

El programa elige la solución que produjo el menor costo, en este caso [1,4], que es una solución incorrecta. Para esta solución, el programa seleccionó el siguiente conjunto de reglas:

- $c_{k,1} = T_1(c_{k,0})$, donde T_1 es la transformación "rotar 45°".
- $c_{k,0} = T(c_{k,2})$, donde T es la transformación identidad.
- $c_{k,1} = T_2(c_{k,0})$, donde T_2 es la transformación "escalar al 15.1% y rotar 45°".
- $c_{k,0} = T(c_{k,3})$, donde T es la transformación identidad.
- $c_{k,0} = T(c_{k+1,3})$, donde T es la transformación identidad.
- $c_{k,1} = T_3(c_{k+1,3})$, donde T_3 es la transformación "escalar al 15.1%".

Al insertar la solución [1,4] en la matriz, este conjunto de reglas produce un traslape positivo de 7, 3 sobrantes, 3 sobrantes positivos, una irregularidad de 2, 6 operandos, 3 transformaciones diferentes de la identidad, 0 negativos verdaderos absolutos, 16 falsos positivos, 0 falsos negativos, 16 falsos positivos absolutos y 0 falsos negativos absolutos. Al insertar estos valores en la Ecuación 2.1 se obtiene un costo de 160.

El programa no logró resolver el problema correctamente utilizando sólo reglas de nivel 2. Sin embargo, eligió la respuesta correcta para la celda *b*. Además, sólo seleccionó reglas que se extienden por filas, de modo que logró reconocer que el problema se resuelve analizándolo por filas.

Nótese que la solución [1,1] produjo un costo de 163, el cual es muy cercano al costo de la solución seleccionada. A pesar de que [1,1] tampoco es la solución correcta, esto sugiere que el programa no está muy seguro de que la solución que eligió sea la correcta. La definición de un concepto de *confianza* a partir de la matriz de costo es un tema que se contempla como trabajo futuro.

Reglas de nivel 2 y 3

Al permitir reglas de nivel 2 y de nivel 3, el programa produjo la matriz de costo de la Tabla 2.21. En este caso el programa seleccionó la solución correcta, [6,4], y seleccionó el siguiente conjunto de reglas para esta solución:

- $c_{k,0} = T(c_{k,2}) \cap T_1(c_{k,1})$, donde T es la transformación identidad y T_1 es "rotar –45°".
- $c_{k,1} = T_2(c_{k,0}) \cap T_2(c_{k,2})^C$, donde T_2 es "escalar al 15.1% y rotar 45°".
- $c_{k,1} = T_3(c_{k,0}) \cap T_3(c_{k,3})$, donde T_3 es "rotar 45°".

Tabla 2.21. Matriz de costo para reglas de nivel 2 y 3.

	1	2	3	4	5	6	7	8
1	117	144	144	120	120	119	124	144
2	126	130	136	126	172	147	186	136
3	115	152	142	118	142	128	156	113
4	136	154	115	126	140	124	126	179
5	126	148	172	106	214	146	203	185
6	104	126	104	**79**	120	126	108	104
7	104	150	130	136	123	150	142	149
8	99	158	133	104	115	129	169	157

Las tres reglas seleccionadas se extienden por filas y son de nivel 3. Al insertar la solución [6,4] en la matriz, este conjunto de reglas produce un traslape positivo de 0, 0 sobrantes, 5 sobrantes positivos, una irregularidad de 2, 6 operandos, 5 transformaciones diferentes de la identidad, 17 negativos verdaderos absolutos, 4 falsos positivos, 14 falsos negativos, 4 falsos positivos absolutos y 14 falsos negativos absolutos. Al insertar estos valores en la Ecuación 2.1 se obtiene un costo de 79. Nótese que se cuenta cada transformación diferente de la identidad, aunque varias de ellas sean iguales entre sí.

Aunque este problema se diseñó utilizando reglas de nivel 4, el programa logró resolverlo correctamente con reglas de nivel 3. Además, en este caso el costo de la solución seleccionada es considerablemente menor al de las demás soluciones, pues la segunda mejor solución es la [8,1] con costo 99.

Reglas de nivel 2, 3 y 4

Al permitir reglas de nivel 2, 3 y 4, el programa produjo la matriz de costo de la Tabla 2.22. El programa también seleccionó la respuesta correcta en este caso, y seleccionó el siguiente conjunto de reglas para esta solución:

- $c_{k,0} = T(c_{k,2}) \cap T_1(c_{k,1}) \cap T_2(c_{k+1,0})^C$, donde T es la transformación identidad, T_1 es "rotar –45°" y T_2 es "escalar al 20.4%".
- $c_{k,1} = T_3(c_{k,0}) \cap T_3(c_{k,2})^C \cap T_3(c_{k,3})^C$, donde T_3 es "escalar al 15.1% y rotar 45°".
- $c_{k,1} = T_4(c_{k,0}) \cap T_4(c_{k,3})$, donde T_4 es "rotar 45°".

Tabla 2.22. Matriz de costo para reglas de nivel 2, 3 y 4.

	1	2	3	4	5	6	7	8
1	92	151	142	126	118	129	123	147
2	123	124	172	128	139	136	124	194
3	100	138	153	134	153	137	124	100
4	132	147	148	133	129	129	122	181
5	117	166	155	128	176	152	176	166
6	117	130	126	**78**	130	128	122	116
7	107	150	147	139	124	137	104	132
8	114	166	138	112	105	121	157	162

Las tres reglas seleccionadas se extienden por filas y las dos primeras son de nivel 4; la tercera es de nivel 3 y es igual a la tercera regla seleccionada en el caso anterior. Al insertar la solución [6,4] en la matriz, este conjunto de reglas produce un traslape positivo de 0, 0 sobrantes, 5 sobrantes positivos, una irregularidad de 2, 8 operandos, 7 transformaciones diferentes de la identidad, 24 negativos verdaderos absolutos, 1 falso positivo, 13 falsos negativos, 1 falso positivo absoluto y 14 falsos negativos absolutos. Al insertar estos valores en la Ecuación 2.1 se obtiene un costo de 78. En este caso, la segunda mejor solución es la [1,1] con costo 92.

La regla esperada

El problema de la Figura 2.19 se diseñó para ser resuelto con una única regla:

- $c_{k,3} = T(c_{k,0}) \cap T_1(c_{k,1}) \cap T(c_{k,2})^C$, donde T es la transformación identidad y T_1 es "rotar –45°".

Esta regla es de nivel 4 y se extiende por filas. Al forzar al programa a seleccionar esta regla para todas las soluciones, se produce la matriz de costo de la Tabla 2.23.

Tabla 2.23. Matriz de costo para la regla esperada.

	1	2	3	4	5	6	7	8
1	233	233	233	182	233	233	233	233
2	233	233	233	182	233	233	233	233
3	233	233	233	182	233	233	233	233
4	233	233	233	182	233	233	233	233
5	233	233	233	182	233	233	233	233
6	167	167	167	**116**	167	167	167	167
7	250	250	250	199	250	250	250	250
8	233	233	233	182	233	233	233	233

El hecho de haber forzado un mismo conjunto de reglas para todas las soluciones provoca que todos los costos fuera de la columna 4 y de la fila 6 sean 233 o 250.

La solución con menor costo es la solución correcta. Al insertar la solución [6,4] en la matriz, la regla esperada produce un traslape positivo de 0, 0 sobrantes, 21 sobrantes positivos, una irregularidad de 1, 3 operandos, 1 transformación diferente de la identidad, 21 negativos verdaderos absolutos, 0 falsos positivos, 0 falsos negativos, 0 falsos positivos absolutos y 0 falsos negativos absolutos. Al insertar estos valores en la Ecuación 2.1 se obtiene un costo de 116.

Bajo estándares humanos, esta regla parece más simple que las inducidas por el programa, pero resulta más costosa porque la Ecuación 2.1 penaliza a los sobrantes positivos, es decir, favorece a los positivos verdaderos sobre los negativos verdaderos. De las 30 figuras que componen el problema, la regla esperada produce sólo 9 positivos verdaderos y 21 negativos verdaderos. El ajuste de la función de costo para reflejar las preferencias humanas es un tema para futuras investigaciones.

2.7 Resumen del capítulo

El lenguaje propuesto para representar los problemas de la RPM está basado la sintaxis del lenguaje Prolog y ofrece las siguientes ventajas:

- Establece las bases de un método para crear descripciones simbólicas a partir de información subsimbólica, utilizando un esquema perceptual que identifica las diferencias y similitudes entre conjuntos de imágenes.
- Es general, pues no define un conjunto finito de figuras básicas.
- Permite separar el procesamiento simbólico del procesamiento de imágenes, pues omite la información visual pero conserva la información espacial y relacional.
- Es eficiente, pues permite representar diagramas utilizando pocos elementos.
- Permite representar de un modo simple los rellenos de figuras y sus propiedades.

La principal desventaja de este lenguaje es que el proceso para representar simbólicamente un problema de la RPM es más complejo que reconocer un conjunto finito de figuras básicas.

A partir de esta representación es posible inducir reglas para resolver la RPM. Con base en la teoría de Pineda (2007), el modelo propuesto genera un espacio de búsqueda que contiene dos tipos de reglas:

- Reglas básicas (de nivel 2), que relacionan pares de celdas de la matriz y son generadas aplicando un esquema de percepción de igualdad a las figuras de un problema.
- Reglas compuestas (de nivel mayor o igual a 3), que relacionan a tres o más celdas de la matriz y son generadas combinando reglas del nivel inmediato inferior con reglas básicas, utilizando un conjunto de heurísticas.

La estructura de estas reglas está basada en la forma normal disyuntiva utilizada en lógica proposicional, y permite representar cualquier fórmula entre conjuntos construida con los operadores de unión, intersección, complemento y transformación. El operador de unión se representa implícitamente mediante la superposición de reglas, mientras que los demás operadores se representan explícitamente.

El modelo de composición de reglas está basado en la técnica de programación lógica inductiva (ILP), que permite inducir relaciones a partir de ejemplos. Esta técnica reduce el problema de la inducción de reglas en la RPM a un problema de búsqueda en un espacio de reglas.

Para seleccionar reglas en este espacio se definieron ocho medidas de desempeño: positivos verdaderos, negativos verdaderos, falsos positivos, falsos negativos y las versiones *absolutas* de las cuatro anteriores. También se definieron tres medidas de complejidad: número de operandos, número de transformaciones diferentes de la identidad, e irregularidad. Estos once atributos se definieron para reglas individuales y para conjuntos de reglas, y permitieron especificar una función de costo para ambos casos.

Los atributos y la función de costo definidos se utilizaron en tres algoritmos para seleccionar reglas individualmente: aleatorio, codicioso y avaro. También se utilizaron en tres algoritmos de búsqueda local para seleccionar conjuntos de reglas: pasos descendentes, búsqueda tabú y recocido simulado. Además se utilizaron en dos algoritmos híbridos: codicioso con pasos descendentes y avaro con pasos descendentes.

Para seleccionar la respuesta de un problema, la maquinaria de inducción y selección de reglas especificada se utilizó de dos formas: generando las celdas faltantes de la matriz, o insertando las posibles respuestas en la matriz.

Las 16 variantes básicas del programa se aplicaron a los 60 problemas de la RPM estándar, y las variantes del algoritmo codicioso produjeron los mejores resultados. La mejor variante fue el algoritmo codicioso con pasos descendentes *b* (insertando las posibles respuestas), la cual logró resolver 54 problemas de la prueba utilizando reglas de nivel 2 y 3.

El modelo propuesto también es capaz de resolver problemas más generales que los de la RPM, por ejemplo:

- Problemas con matrices de tamaño arbitrario, ya sean cuadradas o rectangulares.
- Problemas cuya matriz contiene varias celdas faltantes.
- Problemas que requieren inducir reglas que relacionen a más de tres celdas.
- Problemas que requieren inducir cualquier regla que pueda expresarse mediante los operadores de unión, intersección, complemento y transformación.

Los resultados del modelo proporcionaron varios temas para futuras investigaciones:

- La identificación de la intersección entre figuras, de transformaciones cualitativas y de deformaciones.
- La interpretación de figuras como números.
- La definición de un concepto de *confianza* con base en los costos calculados por el programa.
- El refinamiento del programa para que seleccione reglas más similares a las producidas por las personas.

3 Conclusiones

Este capítulo concluye la exposición de este proyecto. La Sección 3.1 revisa los objetivos presentados en la Sección 1.4. La Sección 3.2 analiza el modelo desarrollado en el contexto de la teoría propuesta por Pineda (2007), que fue presentada en la Sección 1.3. La Sección 3.3 analiza el modelo en comparación con los trabajos anteriores, presentados en la Sección 1.2. La Sección 3.4 aborda cómo la prueba de Raven involucra razonamiento en diferentes niveles de abstracción. Finalmente, la Sección 3.5 expone temas para trabajo futuro que surgieron de este trabajo.

3.1 Revisión de los objetivos

En este proyecto se implementó un programa que resuelve la RPM estándar automáticamente, logrando resolver correctamente 54 de los 60 problemas. Además, el programa es capaz de resolver problemas más generales que los de la prueba.

El programa partió de una representación simbólica de la prueba que se elaboró manualmente. Sin embargo, se establecieron las bases para que esta representación pueda producirse de forma automática en trabajos futuros.

El programa se centró en modelar la inducción de reglas, el aspecto más importante en la resolución de la RPM y un tema poco explorado en trabajos anteriores. Para esto se tomó como base la teoría propuesta por Pineda (2007), como se concluye en la Sección 3.2.

Este proyecto permitió investigar la inteligencia desde varias perspectivas, como la psicométrica, la neurológica, la cultural, la computacional y la del desarrollo cognitivo.

3.2 Esquemas de acción

El presente trabajo estuvo basado en la teoría de Pineda (2007) para la síntesis de conceptos geométricos. En esta teoría se genera un espacio de búsqueda de diagramas, a partir de una figura inicial y un conjunto de esquemas de acción, y es posible sintetizar conceptos geométricos abstractos aplicando principios de conservación en este espacio.

En el presente modelo se genera un espacio de búsqueda de reglas, a partir de las figuras de un problema, un esquema de acción perceptual y un modelo de composición de reglas, y es posible resolver los problemas de la RPM estándar mediante las reglas generadas. El esquema de acción perceptual es crucial para el proceso de inducción, y por sí mismo permitió resolver hasta 38 problemas de la prueba. Al incluirse el modelo de composición de reglas, fue posible resolver hasta 54 problemas.

El modelo propuesto difiere de la teoría de Pineda en dos aspectos principales:

1. En la teoría de Pineda, el espacio de búsqueda se genera mediante esquemas de acción motrices y perceptuales, mientras que en el modelo propuesto sólo se utiliza un esquema de acción perceptual. Esto se debe a que en la resolución de la RPM no se busca generar nuevos diagramas, sino inducir nuevas descripciones (reglas) que representen la lógica de los diagramas existentes.
2. En la teoría de Pineda, los esquemas de acción se aplican sucesivamente para generar el espacio de búsqueda, mientras que en el modelo propuesto se combinan las descripciones producidas por el esquema perceptual para generar el espacio de búsqueda.

De este modo, ambos modelos exploran aspectos complementarios del proceso de descubrimiento de conceptos: por un lado, la aplicación secuencial de esquemas motrices y perceptuales, y por otro lado, la combinación de esquemas perceptuales.

El presente trabajo también permitió explorar cómo los conceptos sintetizados mediante la teoría de Pineda pueden utilizarse en otros procesos de inferencia. En el modelo propuesto, las descripciones inducidas por el esquema perceptual son utilizadas en varias formas:

- Para tratar de explicar las figuras presentes en la matriz (abducción).
- Para tratar de deducir qué figuras deben estar presentes en una celda (deducción).
- Para tratar de describir a las figuras presentes en la matriz de la forma más simple posible (optimización).

3.3 Comparación con trabajos previos

Al final de la Sección 1.2 se compararon los trabajos previos que abordaron la resolución automática de la RPM. En esta sección se complementa este análisis incluyendo el modelo propuesto en el presente trabajo.

Representación

En este trabajo se partió de una representación simbólica y elaborada a mano de la RPM, como en la mayoría de los trabajos anteriores. Sin embargo, el lenguaje de representación propuesto no utiliza figuras predefinidas, sino que explora cómo pueden generarse representaciones simbólicas a partir de información subsimbólica, utilizando un esquema perceptual que compara pares de imágenes.

Esta comparación por pares fue identificada en los experimentos de Carpenter et al. (1990), al analizar el movimiento ocular de las personas que resuelven la RPM. Además, la teoría propuesta por Doumas et al. (2008) también resalta la centralidad de la comparación en el descubrimiento de relaciones estructuradas a partir de información no estructurada.

Por otro lado, tres de los trabajos previos utilizaron representaciones jerárquicas para organizar a las figuras de un problema en varios niveles (como líneas, figuras y grupos). En este trabajo, el esquema de comparación se encarga de agrupar o separar figuras, pues realiza una segmentación basada en comparaciones que agrupa a las figuras que aparecen juntas en todos los diagramas de un problema.

Razonamiento por analogía

Como en los trabajos anteriores, el modelo propuesto incluyó procesos para realizar razonamiento por analogía:

- Procesos para encontrar las correspondencias entre los elementos de la matriz, es decir, para identificar a las figuras gobernadas por una misma regla.
    - Los trabajos anteriores asumían que la matriz debía procesarse por filas o por columnas, usaban analogías, relaciones o reglas predefinidas, relacionaban figuras con el mismo nombre o relacionaban figuras sobrantes.
    - El modelo propuesto sólo asume correspondencia entre las figuras con el mismo nombre en celdas diferentes de la matriz, sin importar si estas celdas se encuentran en la misma fila o columna.
- Procesos para comparar los elementos correspondientes e identificar patrones de variación.
    - Los trabajos anteriores comparaban a los elementos correspondientes para identificar o inducir patrones de variación.
    - El modelo propuesto compara a las figuras con el mismo nombre en celdas diferentes de la matriz para inducir un conjunto de reglas básicas. Las reglas compuestas se generan a partir de las reglas básicas, sin necesidad de volver a comparar figuras o celdas.
- Procesos para generalizar patrones de variación.
    - Los trabajos anteriores comparaban los patrones de variación identificados para definir patrones generales (reglas). Además, varios trabajos utilizaron un conjunto de patrones generales predefinidos.
    - El modelo propuesto representa a todas las reglas inducidas en una forma generalizada, pues las reglas se extienden por filas o por columnas y relacionan a celdas enteras, no a figuras individuales.

El modelo propuesto incluyó un proceso adicional a los anteriores: la selección de las reglas más relevantes para resolver un problema. Los trabajos previos evitaron este problema de tres formas:

- Predefiniendo un conjunto de reglas o un conjunto de relaciones entre las celdas de la matriz, lo cual limita de forma importante el tipo de reglas que el programa es capaz de identificar. Todos los trabajos previos definieron uno o los dos elementos anteriores.
- Asumiendo una relación de orden entre las reglas predefinidas, de modo que unas siempre se prefirieren sobre otras: Carpenter et al. (Betteraven), Lovett et al. (2010) y Cirillo et al.
- Asumiendo que todas las reglas que se cumplen en la matriz son igualmente relevantes, es decir, la relevancia de una regla es un valor verdadero/falso: Carpenter et al. (Fairaven), Lovett et al. (2007), Kunda et al. (método fractal) y Rasmussen et al.

El único programa que abordó una forma de selección de reglas fue el del método afín de Kunda et al., pues selecciona a la transformación afín que se ajusta mejor a las celdas de la matriz, de acuerdo al concepto de similitud visual definido en la Ecuación 1.1. En este caso, la relevancia de una transformación es un valor real en el intervalo [0,1].

En el presente trabajo, la relevancia de una regla o conjunto de reglas es un valor entero en el intervalo $[0,n]$, donde n depende del número de figuras en el problema, del tamaño de la matriz, del tamaño máximo permitido para el conjunto de reglas seleccionadas, del nivel máximo permitido para las reglas y de la función de costo definida en la Ecuación 2.1.

El modelo propuesto reafirmó la importancia de la analogía para inducir relaciones y para resolver la RPM. Además mostró que el proceso para representar simbólicamente los problemas de la prueba también puede verse como un problema de analogía, pues requiere la identificación de correspondencias y la comparación, así como la identificación y selección de patrones generales (en este caso, clases de figuras). Esto hace referencia a la observación de Hofstadter y otros de que incluso los actos mentales más abstractos y sofisticados tienen una profunda semejanza con la percepción (véase Forbus et al., 1998).

Inducción de reglas

Varios de los trabajos previos utilizaron reglas generales predefinidas y resolvieron la RPM mediante deducción en lugar de inducción. Por otro lado, los trabajos que sí modelaron la inducción de reglas obtuvieron un bajo desempeño en la prueba estándar:

- En Lovett et al. (2007) se generan reglas a partir de mapeos del sistema SME y se logró resolver 22 problemas.
- En Kunda et al. (método fractal) se generan reglas a partir de transformaciones fractales y se logró resolver 32 problemas.
- En Rasmussen et al. se generan reglas a partir de transformaciones vectoriales y no se reporta el desempeño del programa, pero el modelo sólo es capaz de relacionar pares de celdas en la misma fila, por lo que su desempeño fue probablemente bajo en las últimas dos secciones de la prueba.

El presente trabajo modeló la inducción de reglas con base en dos elementos:

- Un esquema de percepción de igualdad, que permite generar reglas básicas a partir de las figuras de un problema, y que está basado en el concepto de esquema perceptual definido en la teoría de Pineda (2007).
- Un esquema de composición de reglas, que permite generar reglas compuestas a partir de reglas básicas, y que está basado en la forma normal disyuntiva utilizada en lógica proposicional, y en la técnica de programación lógica inductiva (ILP).

El modelo propuesto es capaz de inducir explícitamente reglas que relacionen a más de dos celdas de la matriz, mientras que los tres trabajos mencionados sólo relacionan pares de celdas. Los resultados obtenidos muestran que la capacidad de inducir reglas de nivel 3 es crucial para obtener un alto puntaje en la RPM estándar, pues mientras el programa logró resolver 54 problemas con reglas de nivel 2 y 3, sólo resolvió hasta 38 problemas con reglas de nivel 2. Los trabajos que utilizaron reglas predefinidas y obtuvieron altos puntajes también eran capaces de relacionar tres celdas de la matriz. El presente trabajo es capaz de inducir reglas que relacionen cualquier número de celdas en cualquier ubicación de la matriz.

El modelo de inducción propuesto comparte elementos con la teoría para el descubrimiento de conceptos relacionales propuesta por Doumas et al. (2008), en la cual:

- La identificación de correspondencias, la comparación y la analogía son centrales para la inducción de relaciones.
- Las relaciones se inducen a partir de ejemplos siguiendo un proceso de refinamiento.
- Las relaciones que involucran a dos o más entidades son construidas combinando relaciones que involucran a una sola entidad.

Aprendizaje

Por cuestiones de tiempo, el modelo propuesto no consideró los efectos del aprendizaje al resolver la RPM, pero este tema se discute como trabajo futuro en la Sección 3.5.

Selección de la respuesta

Como se mencionó en el análisis de la Sección 1.2, los trabajos previos utilizaron dos estrategias diferentes para seleccionar la respuesta de un problema: generar la celda faltante o insertar las posibles respuestas en la matriz.

En este trabajo se implementaron ambas estrategias y la segunda produjo mejores resultados. Sin embargo, la primera se implementó sin permitirle al programa reconsiderar sus hipótesis; además, la segunda requirió un mayor tiempo de procesamiento porque realiza el proceso de inducción una vez para cada posible respuesta del problema. De acuerdo a los experimentos de Carpenter et al. (1990), las personas que obtienen mejores puntajes en la prueba utilizan la primera estrategia, de modo que el refinamiento de la primera estrategia es un tema que se contempla como trabajo futuro.

Resultados

La Figura 3.1 compara los resultados de la mejor variante del programa con los resultados de los trabajos previos que han resuelto la RPM estándar.

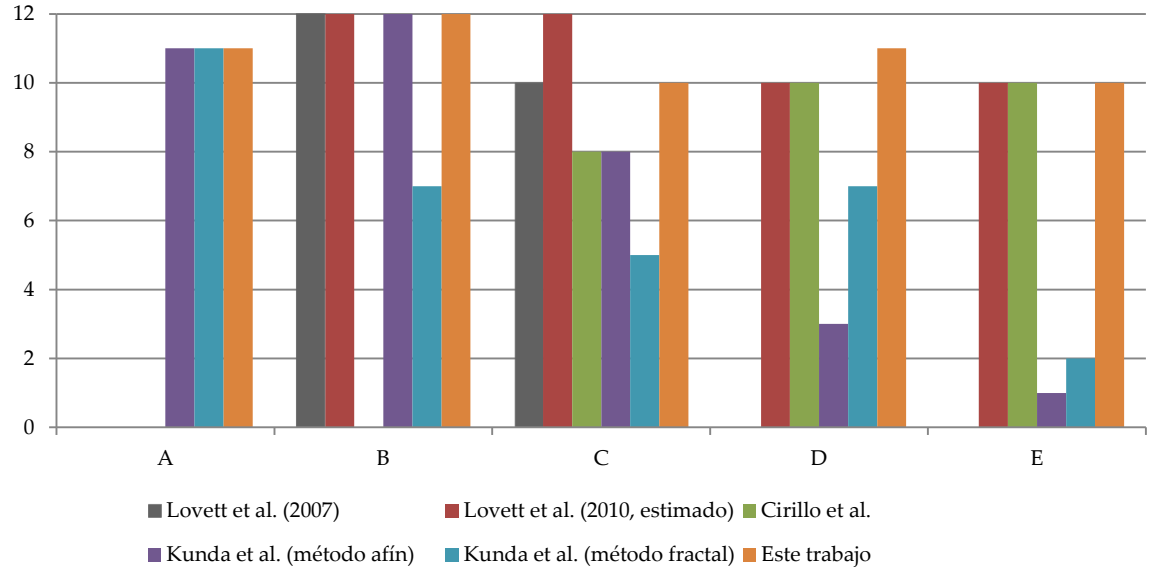


Figura 3.1. Resultados de los programas que han resuelto la prueba estándar[14].

[14] Para una gráfica imprimible en blanco y negro, consulte el Apéndice A.

El programa desarrollado obtuvo el mayor puntaje total al resolver 54 problemas, seguido del programa de Lovett et al. (2010) que resolvió 44. Al considerar sólo a las secciones B a E, el programa desarrollado resolvió un problema menos que el programa de Lovett et al. (2010), el cual, sin embargo, no modeló la inducción de reglas.

Por lo tanto, el programa alcanzó dos metas que ninguno de los programas anteriores cumplió simultáneamente:

- Modelar el proceso de inducción de reglas.
- Obtener un alto puntaje en todas las secciones de la prueba, incluyendo la sección A.

Además logró resolver problemas más generales que los de la RPM, como problemas con matrices de cualquier tamaño o con múltiples celdas faltantes.

3.4 Niveles de abstracción

Los programas desarrollados para resolver la RPM permiten analizar cómo esta prueba involucra razonamiento en diferentes niveles de abstracción. De acuerdo con Pineda (2007), algo es *concreto* si se refiere a entidades, propiedades o relaciones específicas, y es *abstracto* si se refiere a clases de estos objetos. De este modo, la resolución de cada problema puede verse como una progresión desde representaciones concretas hacia representaciones más abstractas:

1. *Imagen sin procesar*: En el nivel más concreto, un problema es una imagen en la cual no se han identificado celdas, figuras ni objetos de ninguna clase.
2. *Problema RPM*: A continuación, es necesario identificar a la matriz del problema, sus celdas, la o las celdas faltantes y las posibles respuestas. Esto requiere establecer una correspondencia entre la imagen del problema y un concepto abstracto de *problema RPM*, el cual contiene a los elementos que debe tener un problema, y engloba a todos los problemas de la prueba como casos específicos.
3. *Figuras*: Las celdas y las posibles respuestas deben compararse entre sí para identificar a las figuras que componen al problema. Esto crea un conjunto de figuras abstractas que permiten segmentar a cada celda y cada respuesta en diferentes objetos. Por ejemplo, al comparar las dos celdas superiores de la matriz en la Figura 3.2, se crea una figura abstracta *flecha curva*, que engloba a las cinco flechas curvas del problema como casos específicos.
4. *Reglas específicas*: Al comparar a las figuras en diferentes celdas de la matriz, se inducen reglas que permiten explicar las variaciones entre estas figuras. Por ejemplo, al comparar las celdas superiores en la Figura 3.2, se induce una regla *flecha curva y grande hacia la izquierda se convierte en flecha curva y pequeña hacia la derecha*.
5. *Reglas generales*: Para resolver un problema es necesario convertir las reglas específicas en reglas generales, las cuales permiten agrupar a las reglas específicas y seleccionar la mejor respuesta para el problema. Por ejemplo, en la Figura 3.2 es necesario identificar la regla general *reducir tamaño y reflejar en el eje Y*, la cual incluye como un caso específico a la regla descrita en el punto 4, y puede aplicarse a la celda inferior izquierda de la matriz para generar la celda faltante.

Como se ha visto a lo largo de este trabajo, la identificación de correspondencias, la comparación, la analogía, la abstracción y el manejo de jerarquías de objetivos son esenciales en este proceso.

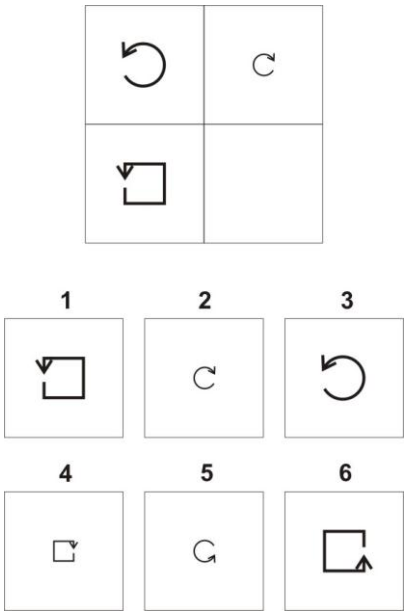


Figura 3.2. Problema con matriz de 2x2.

3.5 Trabajo futuro

Representación

El programa desarrollado no procesa imágenes directamente, sino que parte de una representación simbólica y elaborada a mano de la RPM. Sin embargo, el programa puede extenderse para producir esta representación de forma automática a partir de imágenes escaneadas de la prueba.

En la Sección 2.1 se presentaron ideas para realizar este proceso mediante un esquema de segmentación basada en comparaciones, y utilizando el programa desarrollado en este trabajo para definir la mejor forma de representar un problema.

Por otro lado, el lenguaje de representación también puede extenderse en varias formas, por ejemplo:

- Para representar transformaciones cualitativas, la intersección de figuras y la deformación de figuras.
- Para mejorar la representación de rellenos.

Razonamiento por analogía

Un tema interesante es que la analogía puede aplicarse en varias partes del proceso de resolución de la RPM:

- Para representar simbólicamente los problemas.
- Para inducir reglas entre figuras específicas de la matriz.
- Para inducir reglas generales comparando reglas específicas.

En el presente trabajo, el primer punto se realizó manualmente. Además, los últimos dos se combinaron en uno solo, ya que las reglas inducidas entre figuras se representan directamente en una forma generalizada, que relaciona a celdas completas y que extiende a las reglas por filas o por columnas. Esta forma generalizada es una de las suposiciones del modelo presentado, la cual podría suprimirse aplicando un proceso de analogía al conjunto de reglas específicas identificadas.

Por otra parte, el modelo de selección de reglas puede extenderse y refinarse en diversas formas:

- Utilizando algoritmos de búsqueda y clasificación adicionales a los implementados, como algoritmos genéticos o redes neuronales, los cuales pueden partir del conjunto de atributos definidos en este trabajo para comparar reglas y conjuntos de reglas.
- Así mismo, pueden definirse nuevos atributos para la selección de reglas, considerando, por ejemplo, no sólo cuántas figuras son cubiertas por una regla, sino cuáles son estas figuras y si pertenecen a una misma categoría.
- El proceso de selección también puede refinarse para elegir reglas más simples de acuerdo a estándares humanos, como se vio en la Sección 2.6.

Inducción de reglas

El modelo propuesto puede extenderse para que sea capaz de inducir nuevos tipos de reglas, como las que requieren la interpretación de figuras como números, la identificación de la intersección entre figuras, el manejo de transformaciones cualitativas, la identificación de deformaciones de figuras, o la identificación de progresiones numéricas. Nótese, sin embargo, que la RPM está diseñada como una prueba de razonamiento no verbal que no requiere el uso de habilidades numéricas avanzadas, lo cual permite que la RPM pueda aplicarse a personas con diferentes antecedentes culturales.

Aprendizaje

Como se expuso en el análisis de la Sección 1.2, el aprendizaje juega un papel importante cuando las personas resuelven la RPM. En el modelo propuesto, el aprendizaje puede incluirse en varios de los procesos involucrados, desde el proceso para identificar a las figuras que componen a un problema, hasta el proceso para seleccionar a las mejores reglas.

Selección de la respuesta

El programa desarrollado puede mejorarse de modo que la estrategia de generar la celda faltante sea capaz de corregir sus hipótesis. En Carpenter et al. (1990) se afirma que esta estrategia es utilizada por las personas que obtienen puntajes altos en la prueba. Además, este método consume menos recursos que el de insertar todas las posibles respuestas en la matriz, en especial cuando la matriz contiene varias celdas faltantes.

Por otro lado, en la Sección 2.6 se observó que el proceso de selección de la respuesta puede extenderse de modo que el programa no sólo elija una respuesta, sino que indique cuál es el grado

de confianza de su elección. De este modo, el propio programa podría establecer hipótesis sobre cuáles problemas ha resuelto correctamente y cuáles no.

Resultados

Las mejoras descritas en esta sección permitirían incrementar el puntaje final del programa. Sin embargo, otro tema a investigar es cómo se degrada el desempeño del programa al alterar los diversos módulos que lo componen. Estos resultados podrían compararse con los puntajes obtenidos por diferentes personas para establecer hipótesis sobre por qué las personas obtienen diferentes puntajes en la RPM.

Finalmente, una tarea más inmediata para el presente proyecto es aplicar el programa a las otras versiones de la prueba de Raven: la prueba a color y la prueba avanzada.

Referencias

Betjemann, R. S., Johnson, E. P., Barnard, H., Boada, R., Filley, C. M., Filipek, P. A., Willcutt, E. G., DeFries, J. C., Pennington, B. F. (2009). Genetic covariation between brain volumes and IQ, reading performance, and processing speed. Behavior Genetics 40(2), 135–145.

Bratko, I. (2001). Prolog programming for artificial intelligence. Pearson Education.

Bringsjord, S., Schimanski, B. (2003). What is artifcial intelligence? Psychometric AI as an answer. Proceedings of the 18th International Joint Conference on Artifcial Intelligence, 887–893.

Carpenter, P., Just, M., Shell, P. (1990). What one intelligence test measures: A theoretical account of the processing in the Raven Progressive Matrices test. Psychological Review 97(3), 404–431.

Chater, N., Vitányi, P. (2003). Simplicity: A unifying principle in cognitive science?. Trends in Cognitive Sciences 7(1), 19–22.

Cirillo, S., Ström, V. (2010). An anthropomorphic solver for Raven's Progressive Matrices. Department of Applied Information Technology, Chalmers University of Technology, Report No. 2010:096.

Dayan, P., Abbott, L. F. (2001). Theoretical neuroscience: Computational and mathematical modeling of neural systems. MIT Press.

Dorr, G. M. (2008). Segregation's science: Eugenics and society in Virginia. University of Virginia Press.

Eliasmith, C. (2005). A unified approach to building and controlling spiking attractor networks. Neural Computation 17(6), 1276–1314.

Eliasmith, C., Anderson, C. (2003). Neural engineering: Computation, representation, and dynamics in neurobiological systems. MIT Press.

Doumas, L., Hummel, J., Sandhofer, C. (2008). A theory of the discovery and predication of relational concepts. Psychological Review 115(1), 1–43.

Falkenhainer, B., Forbus, K., Gentner, D. (1989). The structure mapping engine: Algorithm and examples. Artificial Intelligence 41, 1–63.

Flynn, J. R. (1999). Searching for justice: The discovery of IQ gains over time. American Psychologist 54(1), 5–20.

Flynn, J. R. (2007). What is intelligence?: Beyond the Flynn effect. Cambridge University Press.

Forbus, K. D., Gentner, D., Markman, A. B., Ferguson, R. W. (1998). Analogy just looks like high level perception: Why a domain-general approach to analogical mapping is right. Journal of Experimental and Theoretical Artificial Intelligence 10, 231–257.

Gayler, R. (2003). Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience. ICCS/ASCS International Conference on Cognitive Science, 133–138.

Geary, D. C. (2005). The origin of mind: Evolution of brain cognition and general intelligence. American Psychological Association.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. Cognitive Science 7, 155–170.

Gentner, D., Holyoak, K. J., Kokinov, B. (Eds.) (2001). The analogical mind: Perspectives from cognitive science. MIT Press.

Gottfredson, L. S. (1997). Why g matters: The complexity of everyday life. Intelligence 24(1), 79–132.

Gottfredson, L. S. (2010). Intelligence and social inequality: Why the biological link? The handbook of individual differences. Wiley-Blackwell.

Huff, D. (1954). How to lie with statistics. Norton.

Hunt, E. (1974). Quote the raven? Nevermore! Knowledge and Cognition, 129–158. Erlbaum.

Jensen, A. R. (1998). The g factor: The science of mental ability. Praeger.

Kunda, M., McGreggor, K., Goel, A. (2010). Taking a look (literally!) at the Raven's intelligence test: Two visual solution strategies. Proceedings of the 32nd Annual Meeting of the Cognitive Science Society.

Kunda, M., McGreggor, K., Goel, A. (2011). Two visual strategies for solving the Raven's progressive matrices intelligence test. Proceedings of the 25th National Conference on AI (AAAI-2011), 1555–1558.

Laureano, A. L., de Arriaga, F. (2000). Reactive agent design for intelligent tutoring systems. Cybernetics and Systems 31(1), 1–47.

Leeuwenberg, E. (1971). A perceptual coding language for visual and auditory patterns. The American Journal of Psychology 84(3), 307–349.

Lovett, A., Forbus, K., Usher, J. (2007). Analogy with qualitative spatial representations can simulate solving Raven's Progressive Matrices. Proceedings of the 29th Annual Conference of the Cognitive Science Society.

Lovett, A., Forbus, K., Usher, J. (2010). A structure-mapping model of Raven's Progressive Matrices. Proceedings of the 32nd Annual Conference of the Cognitive Science Society, 2761–2766.

McGreggor, K., Kunda, M., Goel, A. (2010). A fractal analogy approach to the Raven's test of intelligence. Workshops at the 24th AAAI Conference on Artificial Intelligence.
URL: http://www.aaai.org/ocs/index.php/WS/AAAIW10/paper/view/2025 (consultada el 14 de enero de 2011).

McKinzey, R. K. (2003). Too dumb to die: Mental retardation meets the death penalty. WebPsychEmpiricist.
URL: http://www.wpe.info/papers_table.html (consultada el 30 de julio de 2011).

Meo, M., Roberts, M. J., Marucci, F. S. (2007). Element salience as a predictor of item difficulty for Raven's Progressive Matrices. Intelligence 35, 359–368.

Neisser, U., Boodoo, G., Bouchard, T. J. Jr., Boykin, A. W., Brody, N., Ceci, S. J., Halpern, D. F., Loehlin, J. C., Perloff, R., Sternberg, R. J., Urbina, S. (1996). Intelligence: Knowns and unknowns. American Psychologist 51(2), 77–101.

Neisser, U. (1997). Rising scores on intelligence tests. American Scientist 85, 440–7.

O'Toole, B. I. (1990). Intelligence and behaviour and motor vehicle accident mortality. Accident Analysis & Prevention 22(3), 211–221.

Palmer, S., Rock, I. (1994). Rethinking perceptual organization: The role of uniform connectedness. Psychonomic Bulletin & Review 1(1), 29–55.

Pineda, L. A. (2007). Conservation principles and action schemes in the synthesis of geometric concepts. Artificial Intelligence 171, 197–238.

Rasmussen, D., Eliasmith, C. (2011). A neural model of rule generation in inductive reasoning. Topics in Cognitive Science 3(1), 140–153.

Raven, J. (2000). The Raven's progressive matrices: Change and stability over culture and time. Cognitive Psychology 41, 1–48.

Raven, J. (2002a). Response to Flynn: Searching for justice: The discovery of IQ gains over time. WebPsychEmpiricist.
URL: http://www.wpe.info/papers_table.html (consultada el 13 de julio de 2011).

Raven, J. (2002b). Intelligence, engineered invisibility, and the destruction of life on Earth. WebPsychEmpiricist.
URL: http://www.wpe.info/papers_table.html (consultada el 13 de julio de 2011).

Raven, J., Raven, J. C., Court, J. H. (2004). Manual for Raven's Progressive Matrices and Vocabulary Scales. Pearson Assessment.

Raven, J. (2005). Lethal intelligence. The Psychologist 18(2), 68.

Raven, J. (2011). Spearman on Intelligence. WebPsychEmpiricist.
URL: http://wpe.info/papers_table.html (consultada el 13 de julio de 2011).

Raven, J. C. (1936). Mental tests used in genetic studies: The performances of related individuals in tests mainly educative and mainly reproductive. M.Sc. Thesis, University of London.

Reed, T. E., Jensen, A. R. (1992). Conduction velocity in a brain nerve pathway of normal adults correlates with intelligence level. Intelligence 16, 259–272.

Rushton, J. P., Jensen, A. R. (2005). Thirty years of research on race differences in cognitive ability. Psychology, Public Policy, and Law 11(2), 235–294.

Shaw, P., Greenstein, D., Lerch, J., Clasen, L., Lenroot, R., Gogtay, N., Evans, A., Rapoport, J., Giedd, J. (2006). Intellectual ability and cortical development in children and adolescent. Nature 440(7084), 676–679.

Shönemann, P. H. (1997). Famous artefacts: Spearman's hypothesis. CPC 16(6), 665–694.

Shönemann, P. H. (2005). Psychometrics of intelligence. Encyclopedia of social measurement, Volume 3, 193–201. Elsevier Inc.

Shönemann, P. H., Heene, M. (2009). Predictive validities: Figures of merit or veils of deception? Psychology Science Quarterly 51, 195–215.

Soulières, I., Dawson, M., Samson, F., Barbeau, E. B., Sahyoun, C. P., Strangman, G. E., Zeffiro, T. A., Mottron L. (2009). Enhanced visual processing contributes to matrix reasoning in autism. Human Brain Mapping 30(12), 4082–4107.

Spearman, C. (1904). "General intelligence" objectively determined and measured. American Journal of Psychology 15, 201–293.

Spearman, C. (1924). Some issues in the theory of g (including the law of diminishing returns). Proceedings of the British Association for the Advancement of Science: Section J – Psychology, 174–181.

Spearman, C. (1927). The abilities of man: Their nature and measurement. MacMillan.

Tomai, E., Lovett, A., Forbus, K., Usher, J. (2005). A structure mapping model for solving geometric analogy problems. Proceedings of the 27th Annual Conference of the Cognitive Science Society, 2190–2195.

Tucker-Drob, E. M. (2009). Differentiation of cognitive abilities across the life span. Developmental Psychology 45, 1097–1118.

Tversky, A. (1977). Features of similarity. Psychological Review 84(4), 327–352.

Van der Helm, P. (2007). Structural information theory.
URL: http://www.nici.kun.nl/peterh/doc/sit.html.

Van Leeuwen, M., Peper, J. S., van den Berg, S. M., Brouwer, R. M., Hulshoff Pol, H. E., Kahn, R. S., Boomsma, D. I. (2009). A genetic analysis of brain volumes and IQ in children. Intelligence 37, 181–191.

Verguts, T., De Boeck, P. (2002). The induction of solution rules in Raven's Progressive Matrices test. European Journal of Cognitive Psychology 14(4), 521–547.

Apéndices

A. Gráfica de resultados en blanco y negro

La Figura A.1 muestra los resultados de los seis programas que han resuelto la RPM estándar, incluyendo el presente trabajo, y está diseñada para ser impresa en blanco y negro.

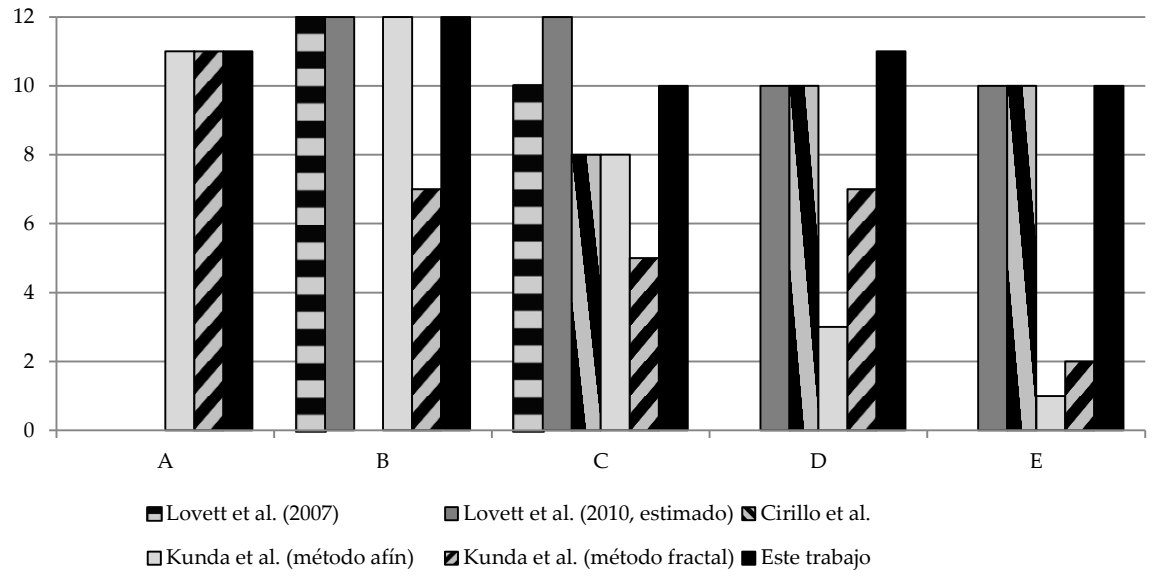


Figura A.1. Resultados de los programas que han resuelto la prueba estándar (blanco y negro).

B. Representación del problema de la Figura 2.19

La Figura B.1 muestra nuevamente al problema de la Figura 2.19. En el lenguaje de representación propuesto en la Sección 2.1, este problema es codificado como se muestra a continuación.

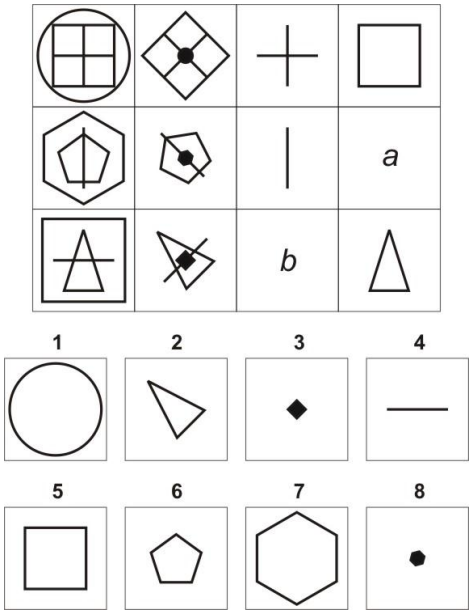


Figura B.1. Problema más general que los de la RPM. La solución es [6,4] (*a* = 6 y *b* = 4).

```
problem( generalProblem,

      matrix(3, [
            diagram([
                  figure([[name,circle], [position,50:50]]),
                  figure([[name,square], [position,50:50]]),
                  figure([[name,vline], [position,50:50]]),
                  figure([[name,vline], [position,50:50], [rotate,90]])
            ]),
            diagram([
                  figure([[name,square], [position,50:50], [rotate,45]]),
                  figure([[name,vline], [position,50:50], [rotate,45]]),
                  figure([[name,vline], [position,50:50], [rotate,135]]),
                  figure([[name,circle], [position,50:50],
                        [scale,0.151:0.151]]),
                  figure([[name,circleFill], [position,50:50]])
            ]),
            diagram([
                  figure([[name,vline], [position,50:50]]),
                  figure([[name,vline], [position,50:50], [rotate,90]])
            ]),
            diagram([
                  figure([[name,square], [position,50:50]])
            ]),

            diagram([
                  figure([[name,hexagon], [position,50:50]]),
                  figure([[name,pentagon], [position,50:50]]),
```

```
                figure([[name,vline], [position,50:50]])
        ]),
        diagram([
                figure([[name,pentagon], [position,50:50],
                        [rotate,45]]),
                figure([[name,vline], [position,50:50], [rotate,45]]),
                figure([[name,hexagon], [position,50:50],
                        [scale,0.151:0.151], [rotate,45]]),
                figure([[name,hexagonFill], [position,50:50]])
        ]),
        diagram([
                figure([[name,vline], [position,50:50]])
        ]),
        diagram([null]),

        diagram([
                figure([[name,square], [position,50:50],
                        [scale,1.352:1.352]]),
                figure([[name,triangle], [position,50:50]]),
                figure([[name,vline], [position,50:50], [rotate,90]])
        ]),
        diagram([
                figure([[name,triangle], [position,50:50],
                        [rotate,45]]),
                figure([[name,vline], [position,50:50], [rotate,135]]),
                figure([[name,square], [position,50:50],
                        [scale,0.204:0.204], [rotate,45]]),
                figure([[name,squareFill], [position,50:50]])
        ]),
        diagram([null]),
        diagram([
                figure([[name,triangle], [position,50:50]])
        ])
]),

answers([
        diagram([
                figure([[name,circle], [position,50:50]])
        ]),
        diagram([
                figure([[name,triangle], [position,50:50],
                        [rotate,45]])
        ]),
        diagram([
                figure([[name,square], [position,50:50],
                        [scale,0.204:0.204], [rotate,45]])
        ]),
        diagram([
                figure([[name,vline], [position,50:50], [rotate,90]])
        ]),

        diagram([
                figure([[name,square], [position,50:50]])
        ]),
        diagram([
                figure([[name,pentagon], [position,50:50]])
        ]),
```

```
            diagram([
                  figure([[name,hexagon], [position,50:50]])
            ]),
            diagram([
                  figure([[name,hexagon], [position,50:50],
                        [scale,0.151:0.151], [rotate,45]])
            ])
      ]),

      solution([6,4])
).
```